Configuring Change Synchronization (Part II)

Practice Overview

This practice has the same target as the target of the previous one, which is configuring the GoldenGate change synchronization. This practice, however, has the following differences than the previous one:

- You do not have to stop the DML operations on the source database.
- You will use a database utility (Data Pump) to perform the initial load process. Generally speaking, this method is faster in big databases than using the GoldenGate initial load methods.

Note: do not get confused by using the Data Pump to create the metadata (as you did in the previous practice) and using it to perform the initial load (as you will do in this practice).

In high level, the procedure will be as follows:

- Delete the Extract, Data Pump, Replicat, and the trails the you have added in the previous practice
- Flush the replicated data in the target database
- Disable the constraints and drop the indexes
- Create a job to imitate continuous DML operations
- Add the new Extract, Data Pump, and Replicate processes
- Start the Extract and Data Pump processes
- Obtain the current source database SCN
- Start the initial load using the `impdp` utility with the FLASHBACK_SCN option
- When the initial load finishes, start the Replicat after the dedicated CSN point
- Create the indexes

Practice Environment

This practice is a continue to the previous practice. It assumes that you have already configured an Extract on the source system and a Replicat on the target system. Therefore, the first part of this practice will be to delete this configuration.

You will learn in this practice, beside its main target, how to drop an Oracle GoldenGate configuration.

Implementing Change Synchronization Option 2

A. Delete the existing Configuration

Because you want to re-create the configuration again from scratch, you need to delete the existing components. However, you do not need to delete the parameter files. You will use the same files again.

1. On the source system, stop and delete the Extract and Data Pump groups.

The GGSCI delete command does not delete the trail files. It removes its checkpoint information from the local checkpoint file in `dirchk`

```
ggsci> DBLogin UserID ogg@db1, Password oracle
ggsci> STOP EXTRACT esrv1
ggsci> DELETE EXTRACT esrv1
ggsci> STOP EXTRACT psrv1
ggsci> DELETE EXTRACT psrv1
```

2. Delete the local trail files.

```
# on ggsrv1 :
ls ./dirdat/es*
rm ./dirdat/es*
```

3. On the target system, stop and delete the Replicat group

```
ggsci> STOP REPLICAT rsrv2
ggsci> DBLogin UserID ogg@db2, Password oracle
ggsci> DELETE REPLICAT rsrv2
```

4. Delete the remote trail files in the target system.

```
# on ggsrv2 :
ls ./dirdat/rt*
rm ./dirdat/rt*
```

B. Flush the data in the target dataset

5. On the target database, disable the HRTRG constraints.

```
SET SERVEROUTPUT ON
exec DBMS_OUTPUT.ENABLE(1000)
DECLARE
 V_SCHEMA VARCHAR2(10) := 'HRTRG';
 V_SQL VARCHAR2(32000);
 CURSOR c_DisableConstraints IS
 SELECT 'alter table '|| OWNER|| '.' ||TABLE_NAME||' disable constraint
'||CONSTRAINT_NAME||' cascade' sqlstatement, TABLE_NAME, CONSTRAINT_NAME
 FROM DBA_CONSTRAINTS WHERE OWNER= V_SCHEMA
 AND TABLE_NAME IN (
'JOB_HISTORY','EMPLOYEES','JOBS','DEPARTMENTS','LOCATIONS','REGIONS')
 ORDER BY TABLE_NAME, CONSTRAINT_NAME;
BEGIN
 FOR r IN c_DisableConstraints LOOP
 BEGIN
  V_SQL := r.sqlstatement;
  EXECUTE IMMEDIATE (V_SQL);
  --DBMS_OUTPUT.PUT_LINE(R.TABLE_NAME ||'-' || R.CONSTRAINT_NAME);
 EXCEPTION
  WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('FAILED: ' || V_SQL);
 END ;
 END LOOP;
END;
/
```

6. Drop the indexes of the replicated HRTRG schema

```
sqlplus / as sysdba
@/home/oracle/scripts/hrtrg_dindexs.sql
```

7. Truncate the tables in the HRTRG schema in the target database.

```
TRUNCATE TABLE HRTRG.JOB_HISTORY;
TRUNCATE TABLE HRTRG.EMPLOYEES;
TRUNCATE TABLE HRTRG.JOBS;
TRUNCATE TABLE HRTRG.DEPARTMENTS;
TRUNCATE TABLE HRTRG.LOCATIONS;
TRUNCATE TABLE HRTRG.REGIONS;
```

C. Create a job to imitate a continuous DML Operations on the source database

The method described in this practice to set up t he change synchronization configuration does not need to stop the DML operations in the source database. As you do not have an application running against the practice environment, let us imitate a simple continuous DML operation.

In this section of the practice, you create a job that will continuously update a table in the HR schema. You will create this table and add it to the replication dataset.

8. Create the following simple table in the source as well as in the target databases.

```
-- in db1
SQL> CREATE TABLE HR.SAMPLE (ID NUMBER(2) PRIMARY KEY, RTIMESTAMP TIMESTAMP ) ;
-- in db2
SQL> CREATE TABLE HRTRG.SAMPLE (ID NUMBER(2) PRIMARY KEY, RTIMESTAMP TIMESTAMP ) ;
```

9. In the source database (`db1`), populate the table with 10 rows.

```
BEGIN
 FOR I IN 1..10 LOOP
   INSERT INTO HR.SAMPLE (ID, RTIMESTAMP) VALUES(I, SYSTIMESTAMP);
 END LOOP;
END;
/
COMMIT;
SELECT ID , TO_CHAR(RTIMESTAMP, ' DD-MM-RR HH24:MI:SS') DTIME FROM HR.SAMPLE;
```

10. Create a procedure that updates the new table.

```
CREATE OR REPLACE PROCEDURE HR.UPDATE_SAMPLE
IS
 N NUMBER(2);
BEGIN
 -- get the most recently updated row
 SELECT MAX(ID) INTO N
 FROM HR.SAMPLE WHERE RTIMESTAMP = (SELECT MAX(X.RTIMESTAMP) FROM HR.SAMPLE X);
 IF N = 10 THEN
  N := 1;
 ELSE
  N := N + 1;
 END IF;
 UPDATE HR.SAMPLE SET RTIMESTAMP= SYSTIMESTAMP WHERE ID=N;
 COMMIT;
END UPDATE_SAMPLE;
/
```

11. Grant the HR user the privilege to create a scheduler job.

```
GRANT CREATE JOB TO HR;
```

12. Unlock the HR account and change its password. Password is case-sensitive by default.

```
ALTER USER HR ACCOUNT UNLOCK;
ALTER USER HR IDENTIFIED BY oracle;
```

13. Create a job that will keep updating that table every 1 minute.

```
conn hr/oracle@db1

BEGIN
 DBMS_SCHEDULER.CREATE_JOB (
 JOB_NAME => 'UPDATE_SAMPLE_JOB',
 JOB_TYPE => 'STORED_PROCEDURE',
 JOB_ACTION => 'UPDATE_SAMPLE',
 START_DATE => SYSDATE,
 REPEAT_INTERVAL => 'FREQ=MINUTELY; INTERVAL=1',
 COMMENTS => 'Keep updating the HR.SAMPLE table');
END;
/

BEGIN
 DBMS_SCHEDULER.ENABLE ('UPDATE_SAMPLE_JOB');
END;
/
```

14. Verify the job is updating the table. Wait for a minute and check out the table contents.

```
SELECT ID , TO_CHAR(RTIMESTAMP, ' DD-MM-RR HH24:MI:SS') DTIME FROM HR.SAMPLE;
```

D. Setup the Oracle GoldenGate processes and trails

In this section, you will add the Extract, Data Pump, local and remote tails, and the Replcat process. You will start the Extracts, but **not** the Replicat process. No need to create the parameter files as you already created them in the previous practice.

15. Add the new table to the Extract parameter file.

For any table to be replicated, GoldenGate needs to add the table supplemental logging. Because you enabled the schema supplemental logging for HR, the supplemental logging of the SAMPLE table will be automatically added. If you did not use the schema-level supplemental logging, you would use the ADD TRANDATA command with the SAMPLE table.

```
ggsci> edit param esrv1

...
Table HR.SAMPLE;
```

16. Verify that the new table (HR.SAMPLE) will be included in the MAP settings in the Replicat parameter file.

```
view param rsrv2
```

17. Add the Extract processes and trails in the source system.

```
Add Extract esrv1, TranLog, Begin Now
Add ExtTrail ./dirdat/es, Extract esrv1
Add Extract psrv1, ExtTrailSource ./dirdat/es
Add RmtTrail ./dirdat/rt, Extract psrv1
```

18. In the **target** system, add the Replicat and link it to its remote trails.

```
Add Replicat rsrv2, ExtTrail ./dirdat/rt, CHECKPOINTTABLE OGG.GG_CHECKPOINT
```

19. In the **source** system, start the Extract and the Data Pump processes

If any of the processes failed to start, check out the `ggserr.log` file

```
start esrv1
info esrv1
start psrv1
info ER *
```

20. Obtain the current source database SCN

```
sqlplus system/oracle@db1
SELECT CURRENT_SCN FROM V$DATABASE ;
```

E. Start the Initial Load

21. In the target system, run the Data Pump Import utility to get the data from the source database. Substitute the value of <SCN> with the number obtained from the previous step.

The code will initialize the whole schema. This is not an issue as the Replicat will apply the changes only on the replicated tables.

```
$ impdp system/oracle@db2 directory=gg_dir schemas=hr remap_schema=hr:hrtrg
remap_tablespace=example:users logfile=hr_db1.log content=DATA_ONLY
flashback_scn=<SCN> network_link=db1link
```

22. Check the output of the command and make sure no error reported.
If the output is too long to view in the window, refer to the log file `hr_db1.log`

F. Start the Replicat

23. After the initial load completes successfully, start the Replicat with the AFTERCSN option.

With this option, the Replicat will skip all the transactions that come before the provided CSN and start applying those which come after that one.

```
start replicat rsrv2, aftercsn <CSN>
info replicat rsrv2
sh ls dirdat
```

G. Create the indexes in the target database

24. Create the indexes of the target dataset.

```
sqlplus HRTRG/oracle
@/home/oracle/scripts/hrtrg_cindexs.sql
```

H. Monitor the change synchronization operation

25. Wait for a few moments, then check out the data in the SAMPLE table in both databases.

```
-- db1
SELECT ID , TO_CHAR(RTIMESTAMP, ' DD-MM-RR HH24:MI:SS') DTIME
FROM HR.SAMPLE ORDER BY 2 DESC;

-- db2
SELECT ID , TO_CHAR(RTIMESTAMP, ' DD-MM-RR HH24:MI:SS') DTIME
FROM HRTRG.SAMPLE ORDER BY 2 DESC;
```

26. Verify that no error reported by the Replicat.

```
tail ggserr.log
```

Note: do not shutdown the systems now. In the next practice you will learn about the proper way of shutting down an Oracle GoldenGate system.

Summary

In this practice you have implemented the online change synchronization from the source database to the target database without disabling the data changes on the source database. You have also used a database utility to perform the initial load.