Using Exceptions Table

Practice Overview

In this practice you will create a table in the target database where the exceptions caught by the Replicat will be saved in it. You will then perform test cases on this functionality.

You will examine the exceptions table functionality first using the `EXCEPTIONSONLY` option, then using the `MAPEXCEPTION` option.

Bug in this practice

While testing this practice in my own test environment, it turns out that a MAP statement with the `EXCEPTIONSONLY` option causes the Replicat process to abend once it applies any operation.

The "`view report rsrv*`" command reports the following error in the end of the report:

```
terminate called after throwing an instance of 'std::out_of_range' what():
vector::_M_range_check
```

This error is not reported in the `ggserr.log` file.

This error needs a *backport* patch from Oracle support to fix it.

A similar bug is reported in Doc ID 2176854.1

Workaround

At this stage, if you face the same error, just restart the process (once or more) and eventually the process will apply the operation and continue working properly.

Using Exceptions Table

A. Create an Exceptions Table for the EXCEPTIONSONLY Option

You start with creating the exceptions table. The table should have the same columns as the columns in the source table. In addition, you can add the following:

- Columns to save the before values. Sometimes it is not necessary to save the before values of all the columns.
- Columns to save the current (target) values.
- Columns to save information about the error.

1. In `db1` (HR schema) and `db2` (HRTRG schema) create the following exceptions table:

```
CREATE TABLE EMPLOYEES_ERRS
( EMPLOYEE_ID       NUMBER(6),
  FIRST_NAME        VARCHAR2(20),
  LAST_NAME         VARCHAR2(25),
  EMAIL             VARCHAR2(25),
  PHONE_NUMBER      VARCHAR2(20),
  HIRE_DATE         DATE,
  JOB_ID            VARCHAR2(10),
  SALARY            NUMBER(8,2),
  COMMISSION_PCT    NUMBER(2,2),
  MANAGER_ID        NUMBER(6),
  DEPARTMENT_ID     NUMBER(4),
  DML_TIME          DATE,
  EMPLOYEE_ID_B     NUMBER(6),
  LAST_NAME_B       VARCHAR2(25),
  SALARY_B          NUMBER(8,2),
  DML_TIME_B        DATE,
  EMPLOYEE_ID_C     NUMBER(6),
  LAST_NAME_C       VARCHAR2(25),
  SALARY_C          NUMBER(8,2),
  DML_TIME_C        DATE,
  RES_DATE          DATE,
  OPTYPE            VARCHAR2(25),
  DBERRUM           VARCHAR2(10),
  DBERRMSG          VARCHAR2(500));
```

2. Edit the parameter file of the Replicat process `rsrv1`. Replace its code with the code below.

In this parameter code, ORACLE_SID is set from within the parameter file using the SETENV statement. In this course environment, because you have only one database and only one Oracle database home, you do not have to set this variable in the parameter file. It is set in this example just for demonstration purpose.

If you have more than one database running in the system, use SETENV parameter to set the ORACLE_SID variable to the database that you want the GoldenGate process to connect to it. The same rule applied for ORACLE_HOME variable.

```
Replicat rsrv1
Include /u01/app/oracle/product/ogg/dirprm/header.mac
SETENV (ORACLE_SID = 'db2')
SETENV (ORACLE_HOME = '/u01/app/oracle/product/12.1.0/db_1')
-- write operations that could not be processed to discard file
RepError (DEFAULT, EXCEPTION)
RepError (DEFAULT2, DISCARD)
DiscardFile ./dirrpt/rsrv1.dsc, Purge
UserIDAlias oggdb1
DBOPTIONS SETTAG 0999
Map HRTRG.EMPLOYEES, Target HR.EMPLOYEES,
 CompareCols (ON UPDATE ALL, ON DELETE ALL),
 ResolveConflict (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
 ResolveConflict (UPDATEROWEXISTS, (DEFAULT, OVERWRITE)),
 ResolveConflict (DELETEROWMISSING, (DEFAULT, DISCARD));
MAP HRTRG.EMPLOYEES, Target HR.EMPLOYEES_ERRS,
EXCEPTIONSONLY,
INSERTALLRECORDS,
SQLEXEC (id qry, query 'select EMPLOYEE_ID, LAST_NAME, SALARY, DML_TIME from
HR.EMPLOYEES where EMPLOYEE_ID = :
p1', PARAMS(p1 = EMPLOYEE_ID )),
COLMAP ( USEDEFAULTS,
    EMPLOYEE_ID_B        = @BEFORE (EMPLOYEE_ID),
    LAST_NAME_B          = @BEFORE (LAST_NAME),
    SALARY_B             = @BEFORE (SALARY),
    DML_TIME_B   = @BEFORE (DML_TIME),
    EMPLOYEE_ID_C        = qry.EMPLOYEE_ID,
    LAST_NAME_C          = qry.LAST_NAME,
    SALARY_C     = qry.SALARY,
    DML_TIME_C   = qry.DML_TIME,
    RES_DATE     = @DATENOW (),
    OPTYPE       = @GETENV ('LASTERR' , 'OPTYPE'),
    DBERRUM      = @GETENV ('LASTERR' , 'DBERRNUM'),
    DBERRMSG     = @GETENV ('LASTERR' , 'DBERRMSG')
    );
Map HRTRG.*, Target HR.*;
```

3. Edit the parameter file of the Replicat process `rsrv2`. Replace its code with the following code:

```
Replicat rsrv2
Include /u01/app/oracle/product/ogg/dirprm/header.mac
RepError (DEFAULT, EXCEPTION)
RepError (DEFAULT2, DISCARD)
DiscardFile ./dirrpt/rsrv2.dsc, Purge
UserIDAlias oggdb2
DBOPTIONS SETTAG 0999
Map HR.EMPLOYEES, Target HRTRG.EMPLOYEES,
 CompareCols (ON UPDATE ALL, ON DELETE ALL),
 ResolveConflict (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
 ResolveConflict (UPDATEROWEXISTS, (DEFAULT, OVERWRITE)),
 ResolveConflict (DELETEROWMISSING, (DEFAULT, DISCARD));
MAP HR.EMPLOYEES, Target HRTRG.EMPLOYEES_ERRS,
EXCEPTIONSONLY,
INSERTALLRECORDS,
SQLEXEC (id qry, query 'select EMPLOYEE_ID, LAST_NAME, SALARY, DML_TIME from
HRTRG.EMPLOYEES where EMPLOYEE_ID = :p1', PARAMS(p1 = EMPLOYEE_ID )),
COLMAP ( USEDEFAULTS,
    EMPLOYEE_ID_B        = @BEFORE (EMPLOYEE_ID),
    LAST_NAME_B          = @BEFORE (LAST_NAME),
    SALARY_B             = @BEFORE (SALARY),
    DML_TIME_B          = @BEFORE (DML_TIME),
    EMPLOYEE_ID_C        = qry.EMPLOYEE_ID,
    LAST_NAME_C          = qry.LAST_NAME,
    SALARY_C     = qry.SALARY,
    DML_TIME_C   = qry.DML_TIME,
    RES_DATE     = @DATENOW (),
    OPTYPE       = @GETENV ('LASTERR' , 'OPTYPE'),
    DBERRUM      = @GETENV ('LASTERR' , 'DBERRNUM'),
    DBERRMSG     = @GETENV ('LASTERR' , 'DBERRMSG')
    );
Map HR.*, Target HRTRG.*;
```

4. Restart the Replicat process in each system and make sure its status is RUNNING.

B. Verify the normal change synchronization is working properly

You will test the change synchronization from `db1` to `db2`, then from the opposite direction.

5. Verify that the SALARY column is the same for EMPLOYEE ID 140 in both `db1` and `db2`.

```
conn hr/oracle
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=140;

conn hrtrg/oracle
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=140;
```

6. In `db1`, update one record in a replicated table.

```
UPDATE EMPLOYEES SET SALARY=SALARY+1000 WHERE EMPLOYEE_ID=140;
COMMIT;
```

7. In `db2`, verify that the update has been replicated. Wait for a few seconds before you try it.

Note: Most likely you will face the bug in this step. Perform the workaround.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=140;
```

C. Testing the Exceptions table functionality

In the following steps, you will perform a test case on the Exceptions table functionality. You will execute a conflicting operation and examine the contents of the Exceptions table.

In this test case, you will insert the records of the same primary key values in both databases (INSERTROWEXISTS conflict).

8. Stop the Data Pump process in both systems.

9. In `db2`, insert a new record in the EMPLOYEES table. Set the LAST_NAME column to ‘DB2’.

```
INSERT INTO HRTRG.EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE,
SALARY, JOB_ID) VALUES (223, 'DB2', 'DB2', 'info@ahmedbaraka.com', TO_DATE('01-01-
2000','DD-MM-YYYY'),10000, 'IT_PROG');
COMMIT;
```

10. In `db1`, insert a new record in the EMPLOYEES table. Set the LAST_NAME column to ‘DB1’.

```
INSERT INTO HR.EMPLOYEES (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, HIRE_DATE,
SALARY, JOB_ID) VALUES (223, 'DB1', 'DB1', 'info@ahmedbaraka.com', TO_DATE('01-01-
2000','DD-MM-YYYY'),10000, 'IT_PROG');
COMMIT;
```

11. Start the Data Pump processes.

12. The conflict that you implemented is INSERTROWEXISTS. You have not defined a resolution for it. Therefore, no change will be made on the record on either of the databases. But the conflict should be reported in the exceptions table.

13. Check the contents of the Exceptions table in both databases.

```
SELECT EMPLOYEE_ID, LAST_NAME, RES_DATE, OPTYPE, DBERRUM, DBERRMSG FROM
EMPLOYEES_ERRS;
```

D. Create an Exceptions Table for the MAPEXCEPTION Option

You have implemented the exceptions table functionality using the EXCEPTIONSONLY option which is used in two different MAP statements for the same mapped table.

In this section of the practice, you will use the MAPEXCEPTION option. This option is used within the same MAP statement that you use in the table mapping.

14. Stop the Replicat in all systems.

15. Drop the exceptions table that you created earlier from each database.

```
DROP TABLE EMPLOYEES_ERRS ;
```

16. In both databases `db1` and `db2`, create the following exceptions table to be used by the MAPEXCEPTION option.

```
CREATE TABLE TRXEXCEPTIONS
(
 REP_GROUP   VARCHAR2(8),
 TABLE_NAME  VARCHAR2(61),
 OPTYPE      VARCHAR2(20),
 ERRTYPE     VARCHAR2(20),
 LOGPOSITION NUMBER,
 COMMITTIMESTAMP TIMESTAMP(6),
 TRAIL_FILE  VARCHAR2(256),
 CDRFAIL     NUMBER,
 CDRSUC      NUMBER,
 CDRDETECT   NUMBER,
 ERRNO       NUMBER,
 DBERRMSG    VARCHAR2(4000)
);
```

17. In `ggsrv1`, replace the contents of the Replicat parameter file with the following code:

```
Replicat rsrv1
Include /u01/app/oracle/product/ogg/dirprm/header.mac
SETENV (ORACLE_SID = 'db2')
SETENV (ORACLE_HOME = '/u01/app/oracle/product/12.1.0/db_1')
-- write operations that could not be processed to discard file
RepError (DEFAULT, EXCEPTION)
RepError (DEFAULT2, DISCARD)
DiscardFile ./dirrpt/rsrv1.dsc, Purge
UserIDAlias oggdb1
DBOPTIONS SETTAG 0999
Map HRTRG.EMPLOYEES, Target HR.EMPLOYEES,
 CompareCols (ON UPDATE ALL, ON DELETE ALL),
 ResolveConflict (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
 ResolveConflict (UPDATEROWEXISTS, (DEFAULT, OVERWRITE)),
 ResolveConflict (DELETEROWMISSING, (DEFAULT, DISCARD));
Map HRTRG.*, Target HR.*,
MAPEXCEPTION (TARGET HR.TRXEXCEPTIONS,
 INSERTALLRECORDS,
 COLMAP (
   REP_GROUP = @GETENV('GGENVIRONMENT', 'GROUPNAME')
 , TABLE_NAME = @GETENV ('GGHEADER', 'TABLENAME')
 , OPTYPE = @GETENV ('LASTERR', 'OPTYPE')
 , ERRTYPE = @GETENV ('LASTERR', 'ERRTYPE')
 , LOGPOSITION = @GETENV ('GGHEADER', 'LOGPOSITION')
 , COMMITTIMESTAMP = @GETENV ('GGHEADER', 'COMMITTIMESTAMP')
 , TRAIL_FILE = @GETENV('GGFILEHEADER', 'FILENAME')
 , CDRFAIL = @GETENV('DELTASTATS','CDR_RESOLUTIONS_FAILED')
 , CDRSUC = @GETENV('DELTASTATS','CDR_RESOLUTIONS_SUCCEEDED')
 , CDRDETECT = @GETENV('DELTASTATS','CDR_CONFLICTS')
 , ERRNO = @GETENV ('LASTERR', 'DBERRNUM')
 , DBERRMSG = @GETENV ('LASTERR', 'DBERRMSG')
         )
);
```

18. In `ggsrv2`, replace the contents of the Replicat parameter file with the following code:

```
Replicat rsrv2
Include /u01/app/oracle/product/ogg/dirprm/header.mac
RepError (DEFAULT, EXCEPTION)
RepError (DEFAULT2, DISCARD)
DiscardFile ./dirrpt/rsrv2.dsc, Purge
UserIDAlias oggdb2
DBOPTIONS SETTAG 0999
Map HR.EMPLOYEES, Target HRTRG.EMPLOYEES,
 CompareCols (ON UPDATE ALL, ON DELETE ALL),
 ResolveConflict (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
 ResolveConflict (UPDATEROWEXISTS, (DEFAULT, OVERWRITE)),
 ResolveConflict (DELETEROWMISSING, (DEFAULT, DISCARD));
Map HR.*, Target HRTRG.*,
MAPEXCEPTION (TARGET HRTRG.TRXEXCEPTIONS,
 INSERTALLRECORDS,
 COLMAP (
   REP_GROUP = @GETENV('GGENVIRONMENT', 'GROUPNAME')
 , TABLE_NAME = @GETENV ('GGHEADER', 'TABLENAME')
 , OPTYPE = @GETENV ('LASTERR', 'OPTYPE')
 , ERRTYPE = @GETENV ('LASTERR', 'ERRTYPE')
 , LOGPOSITION = @GETENV ('GGHEADER', 'LOGPOSITION')
 , COMMITTIMESTAMP = @GETENV ('GGHEADER', 'COMMITTIMESTAMP')
 , TRAIL_FILE = @GETENV('GGFILEHEADER', 'FILENAME')
 , CDRFAIL = @GETENV('DELTASTATS','CDR_RESOLUTIONS_FAILED')
 , CDRSUC = @GETENV('DELTASTATS','CDR_RESOLUTIONS_SUCCEEDED')
 , CDRDETECT = @GETENV('DELTASTATS','CDR_CONFLICTS')
 , ERRNO = @GETENV ('LASTERR', 'DBERRNUM')
 , DBERRMSG = @GETENV ('LASTERR', 'DBERRMSG')
         )
);
```

19. Start the Replicat processes in all the systems.

E. Verify the normal change synchronization is working properly

20. Retrieve the SALARY column value in both databases for EMPLOYEE ID 140.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=140;
```

21. In `db1`, update the SALARY of the record.

```
UPDATE EMPLOYEES SET SALARY=SALARY+1000 WHERE EMPLOYEE_ID=140;
COMMIT;
```

22. In `db2`, verify that the update has been replicated. Wait for a few seconds before you try it.

```
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=140;
```

23. Test the change replication from `db2` to `db1`

```
-- in db2
UPDATE EMPLOYEES SET SALARY=SALARY-1000 WHERE EMPLOYEE_ID=140;
COMMIT;

-- in db1
SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=140;
```

F. Testing the Exceptions table functionality

In the following steps, you will perform a test case on the Exceptions table that you configured for the MAPEXCEPTION option.

In this test case, you will insert records of the same primary key values in both databases (INSERTROWEXISTS conflict).

24. Stop the Data Pump processes in both systems.

25. In db1, insert a new record into the LOCATIONS table.

```
INSERT INTO HR.LOCATIONS SELECT 3300, STREET_ADDRESS, POSTAL_CODE, CITY,
STATE_PROVINCE, COUNTRY_ID from locations where LOCATION_ID = 1000;
COMMIT;
```

26. In db2, insert the same record into the same table.

```
INSERT INTO HRTRG.LOCATIONS SELECT 3300, STREET_ADDRESS, POSTAL_CODE, CITY,
STATE_PROVINCE, COUNTRY_ID from locations where LOCATION_ID = 1000;
COMMIT;
```

27. Start the Data Pump processes.

28. Check the contents of the Exceptions table in both databases.

Observe that the error message does not show the values used in the INSERT statement. If you want to save the new values, you need to create columns in the exceptions table to store the values that you need to save. Then you map them in the COLMAP option.

```
SELECT TABLE_NAME, OPTYPE, DBERRMSG FROM TRXEXCEPTIONS;
```

Summary

With EXCEPTIONSONLY and MAPEXCEPTION options, you can save the exceptions raised by the Replicat into an exceptions table in the database.