GoldenGate Generic Conflict Exception Handler

Edward Whalen
2019-05-10

Recently I was working on Conflict Detection and Resolution (CDR) in a SQL Server GoldenGate environment when I decided to try to create a generic exception handler that I can use for CDR conflicts. The goal was to create an exception handler that can be used for a variety of conflicts without having to create multiple exception tables. The specific case for this handler was to use the same exceptions table for all of my database tables without having extensive customizations. I first created this using a SQL Server to SQL Server environment, then tested the exception handling in Oracle.

By creating a generic exception handler I can apply it to all of my tables without having to create a large number of individual tables. By including the trail file name and RBA I can easily use logdump to find the transaction in question if I need it. Once I have created the generic exception table and a generic Conflict Detection and Resolution (CDR) handler I can run in test in order to determine which tables need additional consideration.

The CDR hander is setup in a macros file I called dirprm\generic.mac and included the following:

```
MACRO #conflict
BEGIN
, COMPARECOLS (ON UPDATE ALL, ON DELETE ALL),
RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, DISCARD)),
RESOLVECONFLICT (INSERTROWEXISTS, (DEFAULT, DISCARD)),
RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, DISCARD)),
RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, DISCARD)),
RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD))
END;
```

Note that I am discarding all of the conflicts. This is only for my initial testing when I am trying to see which tables are getting conflicts. This will not be the end result of conflict resolution. This is very important because this will leave the tables in an inconsistent state. If you already know which system you want to win in a conflict you can add that to the CDR rules.

Again, the goal is to do some initial data collection and determine which tables will potentially receive conflicts and at what rate they are occurring.

For exception handling, the first thing I did was to determine what information that I need in order to be able to find what caused the CDR conflict. Once I had the basics, I would create the exceptions table and exceptions mappings. Some of the basic information that I needed to include was:

- GoldenGate group name
- Group type

- Hostname
- Database Name
- Table Name
- The error that was generated (if any)
- The operation type
- How to find the location in the trail file for the offending statement

I would have liked to log the user as well, but that information is not captured in GoldenGate for SQL Server. In addition, it would have been nice to capture the result of the conflict, but that was not easily achieved. I didn't care to save the actual records of the operation, but I was unable to get the exception handler to work properly without including INSERTALLRECORDS. I did this at the end, but the table columns are not mapped to the generic exceptions table, so they are ignored.

The exceptions handling macro is also included in the file .\dirprm\generic.mac and looks like this:

```
MACRO #exceptions
BEGIN
, TARGET ogg.exceptions          -- on Oracle ggadmin.exceptions
, EXCEPTIONSONLY
, COLMAP (
groupname = @GETENV ('GGENVIRONMENT', 'GROUPNAME')
, grouptype = @GETENV ('GGENVIRONMENT', 'GROUPTYPE')
, hostname = @GETENV ('GGENVIRONMENT', 'HOSTNAME')
, database_name = @GETENV ('DBENVIRONMENT', 'DBNAME')
, table_name = @GETENV ('GGHEADER', 'TABLENAME')
, errno = @GETENV ('LASTERR', 'DBERRNUM')
, dberrmsg = @GETENV ('LASTERR', 'DBERRMSG')
, optype = @GETENV ('LASTERR', 'OPTYPE')
, errtype = @GETENV ('LASTERR', 'ERRTYPE')
, committimestamp = @GETENV ('GGHEADER', 'COMMITTIMESTAMP')
, errortime = @DATENOW()
, ggtrailfile = @GETENV ('GGFILEHEADER', 'FILENAME')
, filerba = @GETENV ('RECORD', 'FILERBA')
)
, INSERTALLRECORDS
END;
```

In order for the exception handling to be used, I had to include the macro file in my replicat parameter file by adding the line:

```
INCLUDE .\dirprm\generic.mac
```

The table creation scripts are for SQL Server:

```
create table ogg.exceptions
(
groupname varchar(10)
```

```
, grouptype varchar(10)
, hostname varchar(40)
, database_name varchar(40)
, table_name varchar(61)
, errno int
, dberrmsg varchar(4000)
, optype varchar(20)
, errtype varchar(20)
, committimestamp datetime
, errortime datetime
, ggtrailfile varchar(40)
, filerba int
);
```

The table creation script for Oracle is:

```
create table ggadmin.exceptions
(
groupname varchar2(10)
, grouptype varchar2(10)
, hostname varchar2(40)
, database_name varchar2(40)
, table_name varchar2(61)
, errno int
, dberrmsg varchar2(4000)
, optype varchar2(20)
, errtype varchar2(20)
, committimestamp timestamp
, errortime timestamp
, ggtrailfile varchar2(40)
, filerba int
);
```

For conflict resolution to work, before images must be taken for update and delete statements. This is placed in the extract file:

```
TABLE soe.* GETBEFORECOLS( ON UPDATE ALL, ON DELETE ALL), FETCHCOLS(*);
```

The CDR and exceptions handlers are setup via macros in the replicat file:

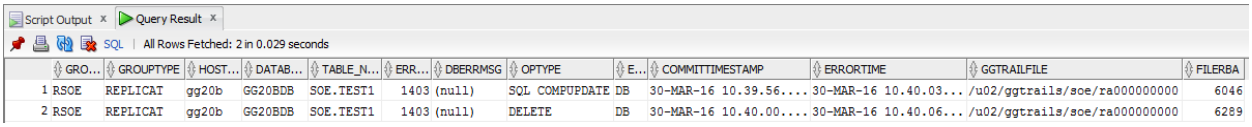
```
MAP SOE.TEST1, TARGET SOE.TEST1 #conflict();
MAP SOE.TEST1 #exceptions();
```

Upon CDR detecting a conflict, the exception table will be populated with enough information to find the point in the trail file where the offending transaction is found.

Results | Messages

	groupname	grouptype	hostname	database_name	table_name	errno	dberrmsg	optype	errtype	committimestamp	errortime	ggtrailfile	filerba
1	REP01	REPLICAT	Ed-Dell	target	dbo.test1	100	[SQL error 100 (0x64)]No data found	DELETE	DB	2016-03-30 22:34:58.883	2016-03-30 22:35:05.000	.\dirdat\1R000070	10195

The Oracle exceptions table is similar but all data might not be populated.



Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.029 seconds

	GRO...	GROUPTYPE	HOST...	DATAB...	TABLE_N...	ERR...	DBERRMSG	OPTYPE	E...	COMMITTIMESTAMP	ERRORTIME	GGTRAILFILE	FILERBA
1	RSOE	REPLICAT	gg20b	GG20BDB	SOE.TEST1	1403	(null)	SQL COMPUPDATE	DB	30-MAR-16 10.39.56....	30-MAR-16 10.40.03...	/u02/ggtrails/soe/ra000000000	6046
2	RSOE	REPLICAT	gg20b	GG20BDB	SOE.TEST1	1403	(null)	DELETE	DB	30-MAR-16 10.40.00....	30-MAR-16 10.40.06...	/u02/ggtrails/soe/ra000000000	6289

This information is very useful and much less difficult to manage than an exception table that partially maps the source table. This exception handler is very generic and can be used with many different tables simultaneously.

Since the trail file and rba are provided, Logdump can easily find this record.

In order to facilitate the parameter file modifications I created the simple shell script that works both under Linux and Cygwin in Windows.

```
for i in `grep MAP $1 | cut -f2 -d" " | cut -f1 -d","`
do
echo "MAP $i, TARGET $i #conflict() ;"
echo "MAP $i #exceptions() ;"
done
```

This shell script will turn this:

```
MAP dbo.test1, TARGET dbo.test1 ;
MAP dbo.test2, TARGET dbo.test2 ;
```

Into an output of the following, which can be cut and pasted into your replicat parameter file:

```
MAP dbo.test1, TARGET dbo.test1 #conflict() ;
MAP dbo.test1 #exceptions() ;
MAP dbo.test2, TARGET dbo.test2 #conflict() ;
MAP dbo.test2 #exceptions() ;
```

To re-iterate, the CDR results in discard which is not appropriate for production environments. However, it is great for testing and determining where CDR is best used.

The exceptions handling is appropriate for all environments.