# Bi-Directional Replication with GoldenGate

## Setup

In its simplest form setup is just the reverse of uni-directional setup. Creating extract and data pump processes on the 12c database and a replicat process on the 10g database.

## Sequences

### Uni-Directional

An explanation of how GoldenGate deals with sequences in a uni-directional setup. You can decide to replicate the actual sequence value by specifying the SEQUENCE parameter in the extract, pump and replicat files.

This causes GoldenGate to check the value of the sequence on the target when it does replication. If the value on the target is lower, then it uses the following formula …

Source\_Value + (Source\_Cache \* Source\_Increment) = Target\_Value

For instance if the Target Sequence Number is 10 and the Source is 11 (Cache 20, Increment 1) then the following will be done

11 + (20\*1) = 31

The target sequence will therefore be changed to 31.
This change is done by a load of “select seq\_name.nextval from dual”.
If you have high cache values this can lead to performance issues.

If you don’t include the sequence parameter then the sequence value on the target database will not be changed by GoldenGate. In this case you would need to manually change them at the time the database was switched over.

### Standard Solutions

It is important that the values for sequences are always different on the 2 databases.
Traditionally this has involved the following solutions …

* Ranges
For this solution you would add 1,000,000 to the sequence next value on one database and hope you won’t use 1,000,000 sequences whilst bi-directional is taking place.

The issue here is that there cannot be big gaps between certain sequences such as Article ID. The value of the Article ID depends on where it appears on the website.
* Odd/Even
There would be odd values on one database and even value on another.
All sequences would be incremented by a value of 2.

For a busy sequence that is updated continuously this can be difficult to setup.
* Concatenation
In some instances you can append a letter at the front of the sequence.
For this to work the sequences values need to be stored as varchars.

All of Mail Online primary keys are stored as number fields so this solution is not feasible.
* Remote Sequences
For this you define the sequence on the source and use it from the target.
A database link and synonym can be used to get the value.

Would require a big update to the application and has the potential to cause performance problems.

### Mail Online Solution

Mix of odd/even and database links.
The business needs to identify the important sequences that cannot get too far out of synchronisation.

* Create database link between source/target and target/source
* For all sequences set increment to 2.
* On target set sequences to be opposite to odd/even on source
* Every 5 minutes compare the important sequence values via database link and move closer by increasing increment or multiple “nextval” commands.

For this solution I would not use GoldenGate sequence replication.
See appendix at the end of this document for full PL/SQL needed to create database objects.

## DDL

It is recommended to nominate a database to perform all DDL on. Conflict Detection and Resolution is not possible with DDL. However it is possible to setup in a bi-directional format. See MOS Document 1521104.1 for further details.

All new tables need have supplemental logging turned on. Because we are not able to use the “Add Schematrandata” command then this has to be included with the “Create Table” command.

A rule would be put in place to ensure that all DDL (which will almost always be a database patch) is only ever executed on the 10g database. This is preferable anyway as we cannot use any 11g or 12c functionality until both databases are at 12c.

## Primary Keys

It is recommended that all tables have primary keys in a uni-directional setup, when it comes to bi-directional this is mandatory. Conflict detection and resolution is impossible if we do not know if a particular insert is allowed or not. Duplicate records are almost certain to occur.

There are a total of 134 tables in the MAILONLINE, RDRCOMMENTS and REGISTRATION schemas that do not have primary keys. Many of these tables are backup tables created when patches or data updates were done. It is hoped that these will be deleted before we go live with bi-directional replication.

The oracle function SYS\_GUID could be used as a primary key for some tables.

## Conflict Detection and Resolution

### Updates

#### Timestamp

A date column is added to all tables called LAST\_UPDATED. This value can be populated either within the code or through the use of a trigger.

The LAST\_UPDATED value before the update on the source is then compared to the current LAST\_UPDATED value on the target. If they are the same then the update is performed, if there is a difference then we have a conflict. You would then normally decide to perform the update if the LAST\_UPDATED value from the target is older/larger than the source.

#### Trusted Source

In this setup you designate one of the database servers as a trusted source.
If there are any conflicts then you always trust this update and overwrite the other.

For this to work you need to capture pre-update values for all columns in the table.
This is the only way to tell if a conflict has occurred.

#### Deltas

If a certain column is updated frequently on both databases then the change needs to be reflected appropriately. An example of this is the Vote Rating for table RC\_COMMENT. At the same time the comment could be voted up and down before GoldenGate has a chance to synchronise the change.
If the Vote Rating was 1 and was changed to 2 on one the 10g database and 0 on the 12c database then the data has diverged. The use of deltas means that both databases will now show the correct value of 1.

#### Row Already Deleted

On occasions an update may be attempted on a record whilst on the other database the record is deleted. In this case it is standard practice to convert the update into an insert.

### Deletes

#### Row Already Deleted

This could be considered an allowable conflict.
If a record is to be deleted then it doesn’t really matter if has been done twice. In this case rules can be put in place to stop GoldenGate abending when it cannot find the record to be deleted.

#### Row Already Updated

However an issue could occur if a record is deleted on the source but at the same time it is updated on the target. Again in this case the business needs to decide how to deal with this situation. Standard practice is to discard the delete.

### Inserts

These aren’t usually considered a conflict.
The main issue is normally around the sequences that the insert may use to generate the primary key. This has already been discussed above.

### Example

In the below example there is a 3 minute lag between the 2 databases.
A particular record is deleted at 10:00am and before this transaction is replicated on the source, it is updated by a different session at 10:02am.

Target

Source

Insert

**Delete (10:00am)**

**Update (10:02am)**

Discard

Because we have setup rules for conflict detection and resolution, GoldenGate decides to discard the delete on the target. Also the update that occurred on the source can now not take place because the record has been deleted. Therefore it is converted into an insert.

The final outcome for this particular record will be as below.

Target

Source

**Update (10:02am)**

**Update (10:02am)**

Written in GoldenGate this would be …

MAP MAILONLINE.ARTICLES, TARGET MAILONLINE.ARTICLES,
GETBEFORECOLS (ON UPDATE ALL, ON DELETE ALL),

MAP MAILONLINE.ARTICLES, TARGET MAILONLINE.ARTICLES,
COMPARECOLS (ON UPDATE ALL, ON DELETE ALL),
RESOLVECONFLICT (UPDATEROWEXISTS, (DEFAULT, USEMAX (MODIFIED\_DATE))),
RESOLVECONFLICT (UPDATEROWMISSING, (DEFAULT, OVERWRITE)),
RESOLVECONFLICT (DELETEROWEXISTS, (DEFAULT, DISCARD)

### Exception Table

When using CDR it is best practice to create an exception table.
This table will track the changes made by the automated resolution processes.

The procedure is …

* Create the exception table under the GGATE schema
* Create a macro which will insert data into this table
* Amend the replicat file so that when an exception occurs, information from that table will be inserted into the exception table.

Full details of the macro and how it is used can be found on the DubeTech wiki at …

<http://dubetech.co.uk/doku.php?id=exceptions>

## Scheduled Database Jobs

There are 9 scheduled jobs in the database that are not owned by SYS / SYSTEM.
Of these, 6 of them are currently enabled.

There are 4 old style jobs (DBA\_JOBS) and 2 scheduler jobs (DBA\_SCHEDULER\_JOBS)

For this particular project it is not a simple choice to just disable them all on one database and enable on the other. Therefore the following should be done …

|  |  |  |  |
| --- | --- | --- | --- |
| **Job Name / Description** | **Job Type** | **Enabled (10g)** | **Enabled (12c)** |
| ce\_queue\_detail.p\_set\_queue\_live | DBA\_JOB | Yes | No |
| p\_set\_polls\_respcounts | DBA\_JOB | Yes | No |
| UPDATE HOROSCOPE SET STATUS\_ID = 4 | DBA\_JOB | Yes | No |
| Refresh MV MV\_LATEST\_ARTICLE\_AUTHOR | DBA\_JOB | Yes | Yes |
| VIDEO\_MAINTENANCE | SCHEDULER\_JOB | Yes | No |
| MO\_NIGHTLY\_CHAIN\_1\_JOB | SCHEDULER\_JOB | Yes | No |

The materialised view refresh is enabled on both databases because the data within it is being excluded from the replication. It was excluded to avoid ORA-01732 (Data manipulation operation not legal).

##

## Compatibility

As we are using different versions of GoldenGate on our 2 databases, things become a little bit more complicated when we transfer the trail files from GoldenGate 12 to 11.

In the Extract file, after the EXTTRAIL location, you need to specify the release.
EXTTRAIL ./dirdat/gr, format release 11.2

In the Data Pump file, after the RMTTRAIL location, you need to specify the release.
RMTTRAIL ./dirdat/gr, format release 11.2

In GoldenGate 12 the length of the trail files was extended to 9 digits.
This is incompatible with GoldenGate 11.

In the GLOBALS file within GoldenGate 12, enter the following line
TRAIL\_SEQLEN\_6D

## Cascaded Deletes

Within the whole MOL database there is only 1 foreign key constraint with a cascaded delete.
This is in the MAILONLINE schema, table USER\_TRANSFER\_ITEM, remote table TRANSFER\_ITEMS.

Any ID deleted from TRANSFER\_ITEMS will also cause records with the same ID to be deleted from USER\_TRANSFER\_ITEM. This causes a problem with replication as now 2 deletes will be done on the same record (Cascade delete plus the replication of the cascade delete).

One solution is to put the following entry into the replicat parameter file …

MAP MAILONLINE.USER\_TRANSFER\_ITEM, TARGET MAILONLINE.USER\_TRANSFER\_ITEM, REPERROR (1403, IGNORE);

Another (which is what was implemented) is to use Conflict Detection and Resolution.
Use: RESOLVECONFLICT (DELETEROWMISSING, (DEFAULT, DISCARD)

## Lag

Lag time is the difference between a transaction being committed on the source database and it being replicated on the target. In a bi-directional setup lag is almost certain to occur and, as mentioned above in the CDR section, rules should be put in place to deal with the issues that will arise.

However in your testing environment you may find as the databases are less busy that there is hardly any lag. You can simulate lag by using the following parameter …

DEFERAPPLYINTERVAL.

For example if you want to defer the replication by 2 minutes you can add the following entry in your replicat parameter file …

DEFERAPPLYINTERVAL 2 MINUTES

## Demonstrations

### Sequences

File: seq\_testing.sql
Increases 5 sequence values by 200.
Then run “mol\_sequences.mol\_sequence\_sync(owner, sequence, min\_sync\_value)” on the other database. This should cause the sequence values to synchronise. Note that caching of 20 will mean that there could still be a gap of up to 40 between the values.

### Updates (Based on Modified Date)

File: article\_update.sql
Use article\_id between 3187000 and 3187999
Updates the same article on the 2 databases at the same time. Run and commit first on the 10g database, then 12c. This example will reject the update on the 10g database because it will have an older timestamp than the 12c update.

Consider an Exception Table

### Updates (Based on Trusted Source)

File: images\_update.sql
Use image\_id of 1234567 (cow) or 1240000 (balls)
This is an example of a table without a modified date column. Therefore in order to deal with conflicts we have to tell GoldenGate which database should be trusted. In this demonstration the 10g database is trusted and therefore the update done on the 12c database is rejected.

Again this example needs an Exception table as data from one database has been lost.

### Updates (Based on Deltas)

File: rc\_comment.sql
Up votes a comment on the 10g database. At the same time down votes the comment on 12c
This example shows how GoldenGate collates both actions and resolves the updates so that both databases are synchronised with the correct number of votes and their overall rating.

### Deletes (Row Missing)

File: QFT\_delete.sql
QUARTZ\_FIRED\_TRIGGERS is a table where a lot of delete operations take place. It is possible that the same delete will take place on both databases at the same time. In this case we are trying to delete a record that has already been deleted. The best practice is to ignore the delete operation and continue.

## Appendix

### PL/SQL for Sequence Solution

The PL/SQL can be found on the DubeTech Wiki at …

<http://dubetech.co.uk/doku.php?id=plsql_sequence>

## Parameter Files

The parameter files used for the 10g and 12c installations can be found on the DubeTech Wiki.
Under the heading “Parameter Files” …

<http://dubetech.co.uk/doku.php?id=goldengate>

## Alerting Script for Checking Errors

This runs as a cron job on all production servers with GoldenGate installed.
The script is stored in $GGHOME/dirsql, acopy is also on the DubeTech Wiki …

<http://dubetech.co.uk/doku.php?id=alert_script>