ACHBANI Mohamed
17-03-2012

Setting up Oracle Golden Gate between 2 Oracle instances (asynchronous)

Requirements:

2 databases GGSRC (source) and GGDEST (destination).
Identical schemas in databases we want to replicate (TEST in our example).
Network connection between two databases.
Ports need to be opened in range 7809 – 7950 in both directions

Preinstallation tasks:

1. Tell database to log more (supplemental logging ~10% of more redo writing) on source side.

```
SQL> alter database add supplemental log data;
```

2. GoldenGate user creation on source and target side

```
$ useradd -d /home/gguser gguser
```

3. I preferred to put GoldenGate database user tables into different tablespace. So create the tablespace for both databases like:

```
SQL>Create
tablespace OGG_DATA datafile '/u02/your_instance_name/ogg_data01.dbf' size
1000M autoextend on next 100M;
```

4. GoldenGate schema creation into source and target database

```
SQL> create user OGG identified by  OGG default tablespace OGG_DATA
temporary tablespace GTMP profile DEFAULT;
alter user OGG QUOTA UNLIMITED ON OGG_DATA;
grant CONNECT to OGG;
grant CREATE SESSION to OGG;
grant RESOURCE to OGG;
grant SELECT ANY TABLE to OGG;
grant ALTER SESSION to OGG;
grant CREATE TABLE to OGG;
grant FLASHBACK ANY TABLE to OGG;
grant SELECT ANY DICTIONARY to OGG;
grant DBA to OGG;
```

(some of the privileges do not make any sense because most of them are covered in roles CONNECT and RESOURCE)
(Never do not give DBA privileges to OGG, I did it because GG doc asked me to give insert update delete privileges for replication schema, I thought later that this was for symmetric GG set up. Then you should give object privileges for certain schema objects)

Installation part:

1**.** Log in or su as gguser (golden gate os user we created before), create directory for GG binaries (OGG – as Oracle Golden Gate is good), unpack your installation pack and that’s all. You have to do this on the both sides.

```
$ mkdir OGG
$ cd OGG
```

Copy you installation pack in there and:

```
$ gunzip ggs_redhat_x64_your_version.tar.gz
$ tar –xvf ggs_redhat_x64_ your_version.tar
```

2. Set up your environment:

Easiest way is to add these 4 lines into your bash_profile.

(Of course edit these lines according to your current environment.)
On the source side:

```
$ export ORACLE_HOME=/data/app/oracle/product/11.2.0
$ export ORACLE_SID=GGSRC
$ export LD_LIBRARY_PATH=/data/app/oracle/product/11.2.0/lib
$ export PATH=$PATH:$ORACLE_HOME/bin
```

On the destination side:

```
$ export ORACLE_HOME=/data/app/oracle/product/11.2.0
$ export ORACLE_SID=GGDEST
$ export LD_LIBRARY_PATH=/data/app/oracle/product/11.2.0/lib
$ export PATH=$PATH:$ORACLE_HOME/bin
```

3. Verify if your main tool ggsci works. In your installation directory execute:

```
 $ ./ggsci
```

If no errors when opening application, then it should work.
That’s all. Installation is done. It’s even a bit weird to describe it, because it’s too simple.

Configuration.

Source side on your ggsci prompt:

```
$ GGSCI (srchost) 1> create subdirs
$ GGSCI (srchost) 1> status all
$ GGSCI (srchost) 1> edit params mgr
```

Your default editor has been opened, insert following lines and save:

```
port 7809
lagreportminutes 5
laginfominutes 1
lagcriticalminutes 2
purgeoldextracts ./dirdat/t*, minkeepdays 2, usecheckpoints

$ GGSCI (srchost) 1> start mgr
$ GGSCI (srchost) 1> status all
```

If manager is running then manager configuration is ok.

```
$ GGSCI (srchost) 1> dblogin userid OGG, password OGG
$ GGSCI (srchost) 1> list tables test.*
```

If you see the list of tables then your configuration is good and you can continue:

```
$ GGSCI (srchost) 1> add trandata TEST.*
$ GGSCI (srchost) 1> info trandata TEST.*
```

Lets create and configure now the extractor process, edit add these files and save:

```
$ GGSCI (srchost) 1> edit params xtst01

extract xtst01
userid OGG, password OGG
discardfile ./dirrpt/xtst01.dsc,purge
reportcount every 15 minutes, rate
exttrail ./dirdat/t1
table TEST.*;

GGSCI (srchost) 1> add extract xtst01, tranlog, begin now
GGSCI (srchost) 1> add exttrail ./dirdat/t1, extract xtst01, megabytes 100
GGSCI (srchost) 1> status all
GGSCI (srchost) 1> start xtst01
```

Extractor created, now we create the data pump process:

```
$ GGSCI (srchost) 1> edit params ptst01

extract ptst01
passthru
rmthost desthost, mgrport 7809
rmttrail ./dirdat/t1
table TEST.*;

$ GGSCI (srchost) 1> add extract ptst01, exttrailsource ./dirdat/t1, begin now
$ GGSCI (srchost) 1> add rmttrail ./dirdat/t1, extract ptst01, megabytes 100, begin
now
$ GGSCI (srchost) 1> status all
```

```
Program    Status     Group     Lag        Time Since Chkpt

MANAGER    RUNNING
EXTRACT    STOPPED    PTST01    00:00:00    00:00:25
EXTRACT    RUNNING    XTST01    00:00:00    00:00:06
```

We are not going to start pump process yet, because destination is not configured.

Target side on your ggsci prompt:

```
$ GGSCI (desthost) 1> create subdirs
$ GGSCI (desthost) 1> edit params mgr
```

```
port 7809
dynamicportlist 7900-7950
lagreportminutes 5
laginfominutes 1
lagcriticalminutes 2
purgeoldextracts ./dirdat/t*, minkeepdays 2, usecheckpoints
```

```
$ GGSCI (desthost) 1> start mgr
$ GGSCI (desthost) 1> status all
$ GGSCI (desthost) 1> view report mgr
```

Manager created and verified, now we are going to create replicat processes. I'll create the replicat with the 5 minutes lag, I just want replication to be 5 minutes behind (testing purpose):

```
$ GGSCI (desthost) 1> edit params rtst01
```

```
replicat rtst01
userid OGG, password OGG
discardfile ./dirrpt/rtst01.dsc, purge
assumetargetdefs
reportcount every 15 minutes, rate
batchsql
deferapplyinterval 5 mins
map TEST.*, target TEST.*;
```

```
$ GGSCI (desthost) 1> add replicat rtst01, exttrail ./dirdat/t1, NODBCHECKPOINT --
CHECKPOINTTABLE=OGG.checkpoint
```

That should be all about configuration , now last two steps, lets start data pump process on source side and replicat process on destination.

```
$ GGSCI (srchost) 1> start ptst01
$ GGSCI (desthost) 1> start rtst01
```

Now just check statuses on both sides with "status all" command. If everything is running and nothing abandoned you can check event logs on both sides and activity:

```
$ view ggsev
$ stats xtst01, totalsonly *, reportrate sec
$ stats rtst01, totalsonly *, reportrate sec
$ send RTST01, status
```

That's all about initial Oracle Golden Gate installation and setup for one asynchronous data stream.