GoldenGate: An Introduction

Patrick Hurley
Accenture Enkitec Group

Keywords:

GoldenGate, logical replication, migration

Introduction

GoldenGate is a product that can move data from a source data store to a target data store. Both data stores are typically, but not limited to being, relational databases.

This data movement is either a regular activity (e.g. data transfer between two operational databases or data feed to a data warehouse) or a one-off activity (e.g. database upgrade or hardware migration).

GoldenGate can move data between many relational databases (e.g. Oracle, SQL Server, DB2, Teradata) and other sources and targets (e.g. Hadoop, Kafka, flat files).

The source and target do not need to be identical, e.g. Oracle to Hadoop, SQL Server to Oracle. The source and/or target can be hosted on-premises or in a public or private cloud.

GoldenGate can perform bi-directional replication between source and target. A key use case for this is to provide a backout option for database migrations, i.e. return to the original database if a problem is encountered on the new database.

GoldenGate can be used to provide Disaster Recovery (although other products would be more appropriate, e.g. Oracle Data Guard) or High Availability (although other products would be more appropriate, e.g. Oracle Real Application Clusters).

This document describes the configuration of GoldenGate replication from an Oracle 11.2 database (named skylark, hosted on server sycamore1) to an Oracle 12.2 database (named penguin, hosted on server poplar1).

The document concludes with some important points about the type of replication (logical replication) used by GoldenGate and other comparable products.

Patch Oracle Databases

Before installing and configuring GoldenGate, check whether source and target databases require patching to support GoldenGate replication. Oracle Support note 1557031.1 (support.oracle.com) lists required and recommended patches for supported Oracle Database versions.

Download GoldenGate

The latest version of GoldenGate can be downloaded from oracle.com/technetwork/middleware/goldengate.

As of November 2017, this is GoldenGate 12.3 (download name: 123010_fbo_ggs_Linux_x64_shiphome.zip).

Introduction to GoldenGate Installation

GoldenGate is typically installed on the same servers as the source and target databases. It can alternatively be installed on its own dedicated server(s). In this document, it is installed on the source (sycamore1) and target (poplar1) database servers, using the same operating system user as the Oracle databases.

GoldenGate installation involves the unzip of the downloaded software and invocation (either interactively or 'silently') of the Oracle Universal Installer.

To show these alternatives, this document describes interactive installation for the source 11.2 database and silent installation for the target 12.2 database. However, either installation could have been performed using either method.

Interactive Installation for an 11g Database

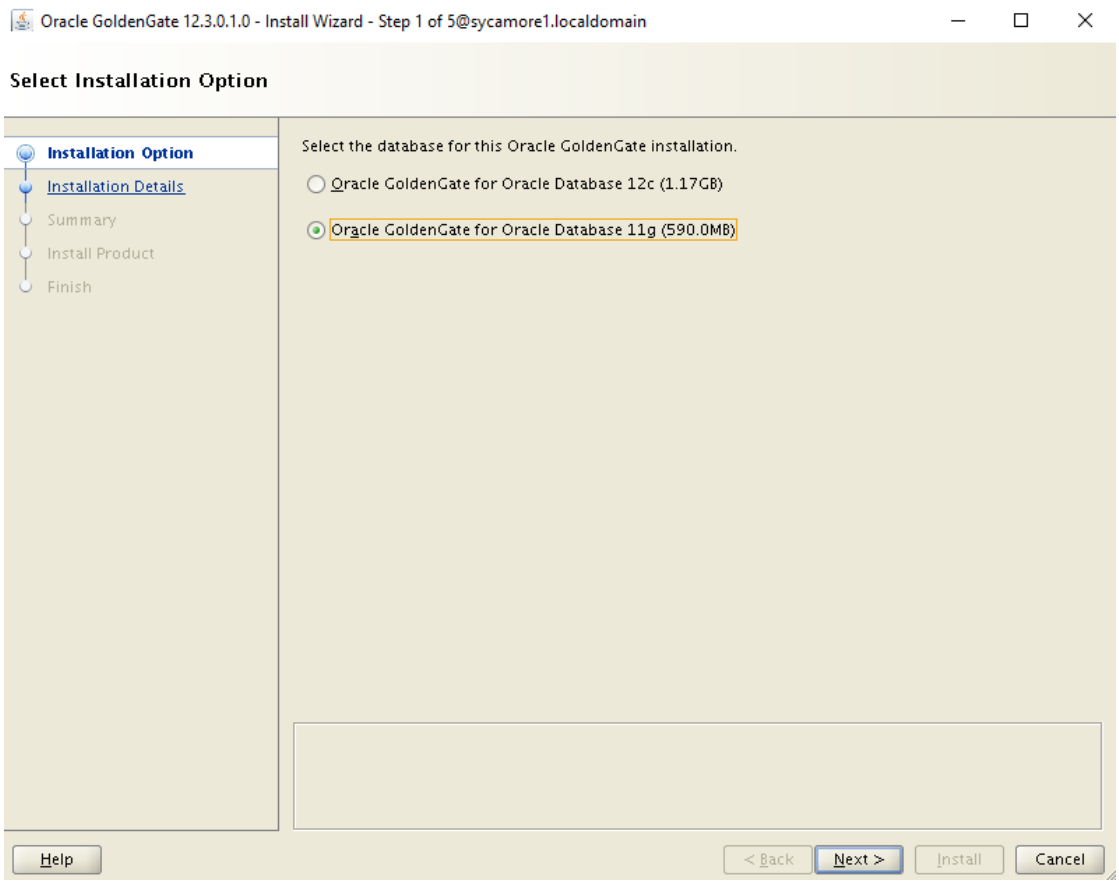
Connect to the source database server as the Oracle database user. Create a directory to host the GoldenGate software, i.e. the Oracle Home for the GoldenGate installation:

```
$ mkdir /u01/app/oracle/product/ggsycamore
```

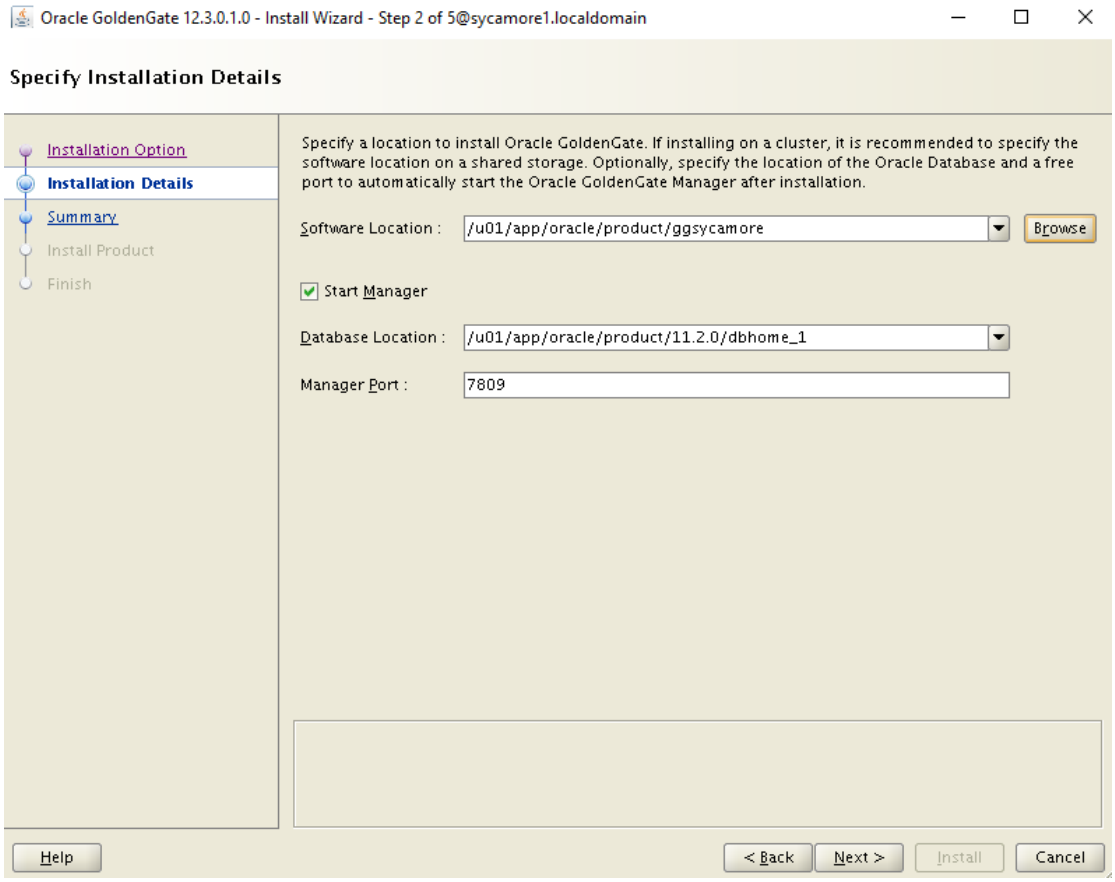
Unzip the installation software into a temporary location, navigate to the directory containing the installer and launch the installer:

```
$ unzip 123010_fbo_ggs_Linux_x64_shiphome.zip
$ cd fbo_ggs_Linux_x64_shiphome/Disk1
$ ./runInstaller
```

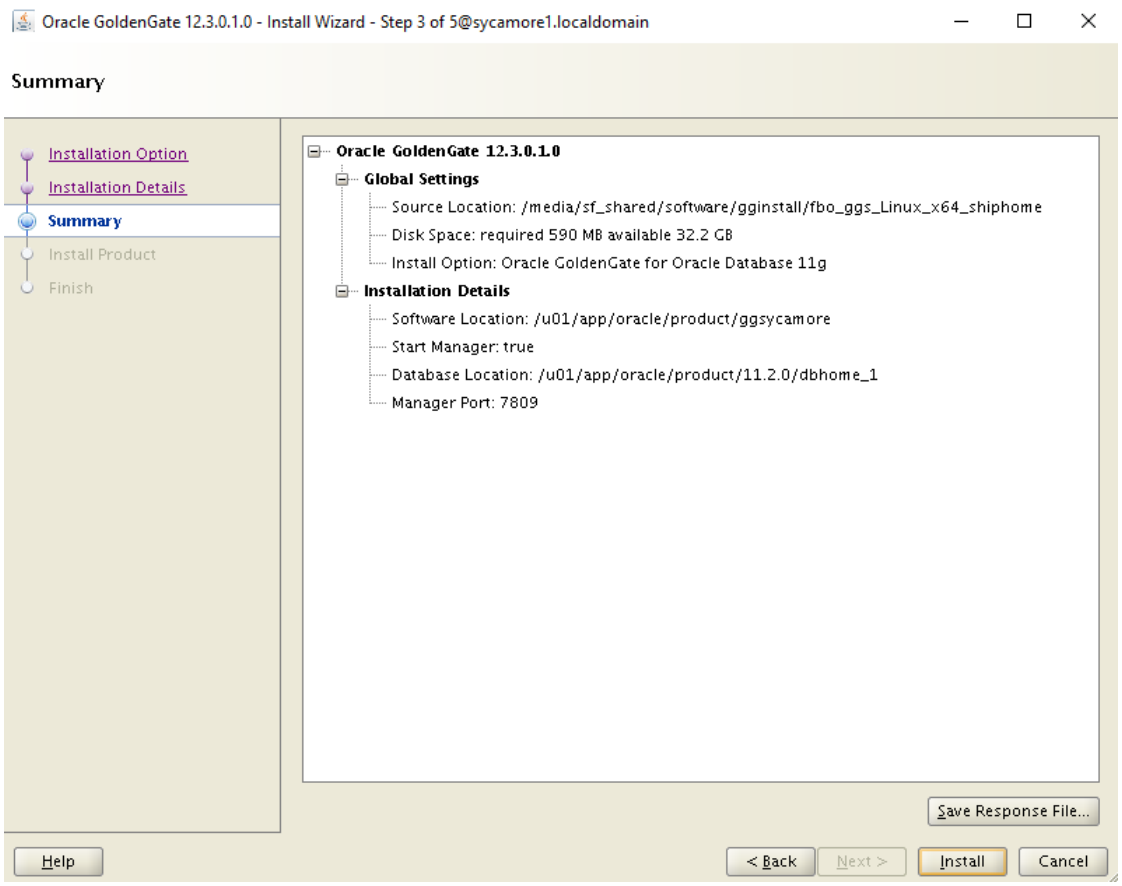
Select the installation option appropriate to the source database version, i.e. 'Oracle GoldenGate for Oracle Database 11g':

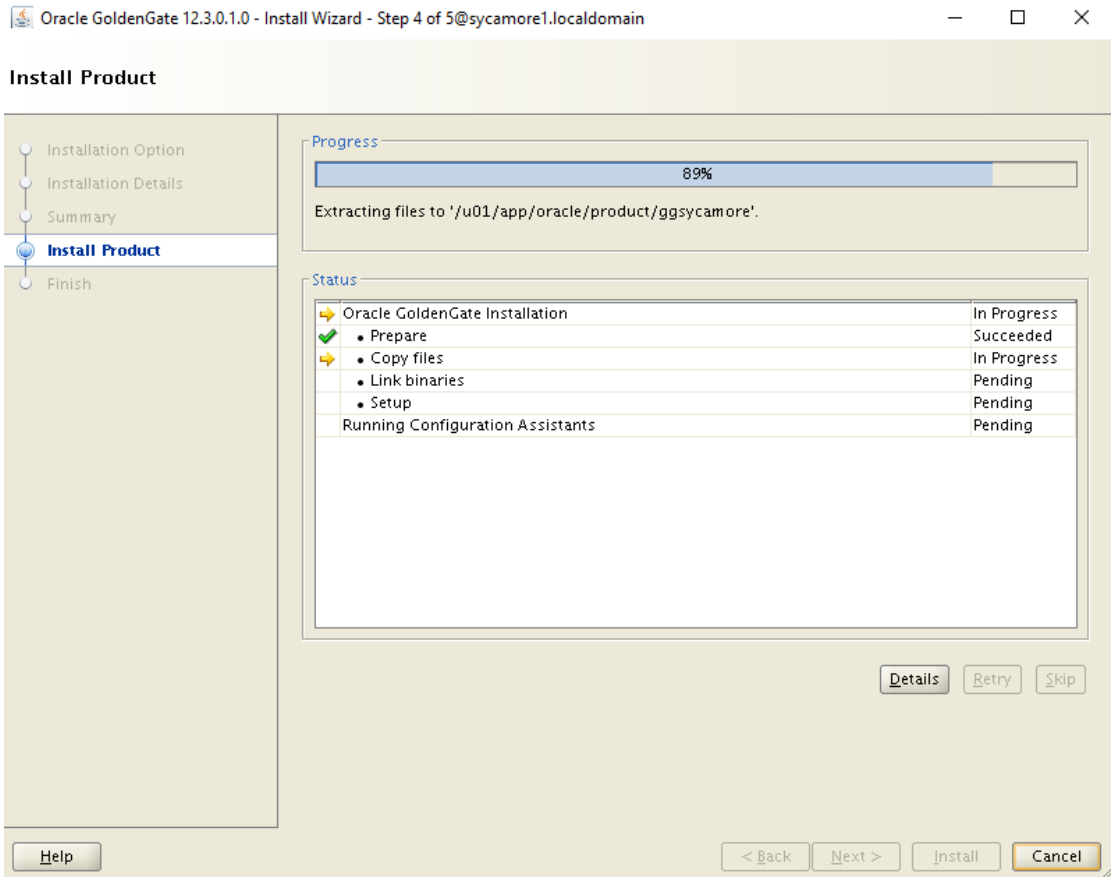


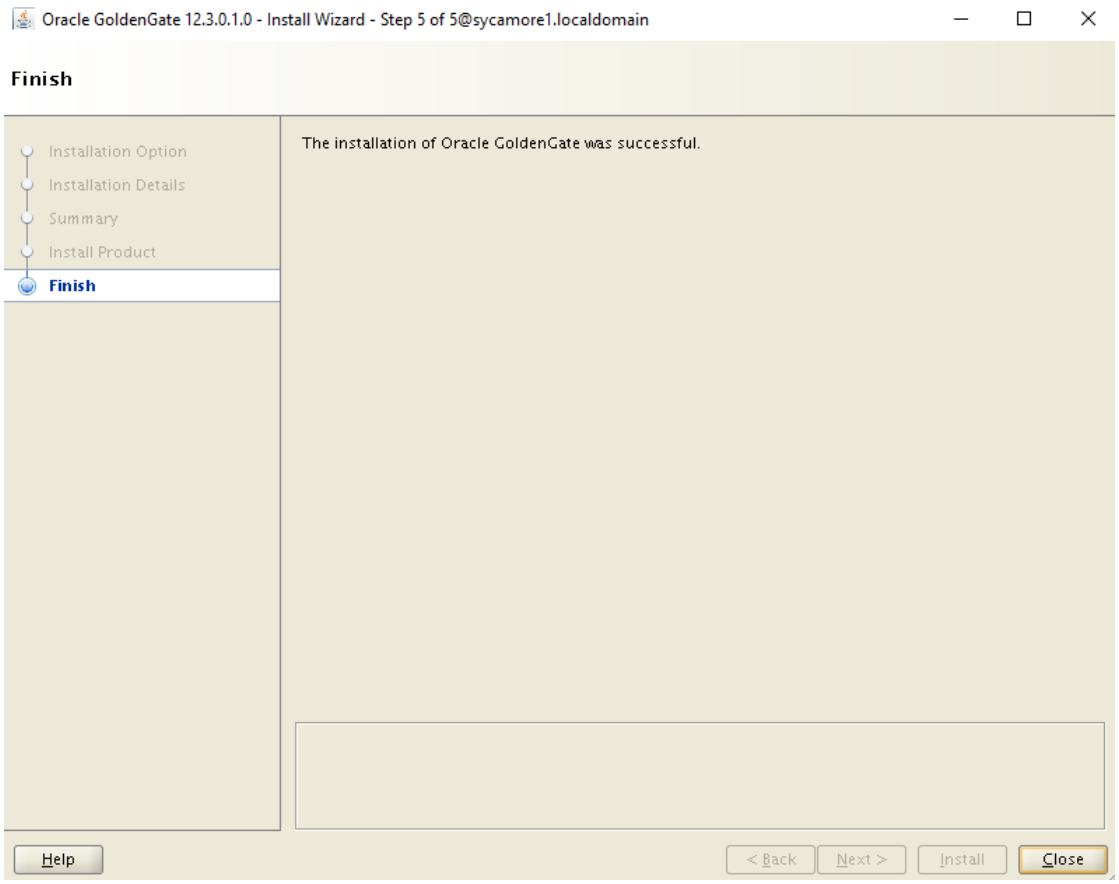
Select a directory in which to store the GoldenGate software (i.e. the directory created above) and the location of the Oracle Home for the source database. Typically leave the GoldenGate Manager Port at the default value, 7809:



Review the installation summary and choose 'Install':







The GoldenGate installation can now be seen referenced in the server's Oracle inventory:

```
$ cat /u01/app/oraInventory/ContentsXML/inventory.xml
```

```
<?xml version="1.0" standalone="yes" ?>
<!-- Copyright (c) 1999, 2017, Oracle and/or its affiliates.
All rights reserved. -->
<!-- Do not modify the contents of this file by hand. -->
<INVENTORY>
<VERSION_INFO>
   <SAVED_WITH>12.2.0.1.4</SAVED_WITH>
   <MINIMUM_VER>2.1.0.6.0</MINIMUM_VER>
</VERSION_INFO>
<HOME_LIST>
<HOME NAME="OraDb11g_home1" LOC="/u01/app/oracle/product/11.2.0/dbhome_1" TYPE="O" IDX="1"/>
<HOME NAME="OraHome1" LOC="/u01/app/oracle/product/ggsycamore" TYPE="O" IDX="2"/>
</HOME_LIST>
<COMPOSITEHOME_LIST>
</COMPOSITEHOME_LIST>
</INVENTORY>
```

Silent Installation for a 12c Database

Connect to the target database server as the Oracle database user. Create a directory to host the GoldenGate software, i.e. the Oracle Home for the GoldenGate installation:

```
$ mkdir /u01/app/oracle/product/ggpoplar
```

Unzip the installation software into a temporary location:

```
$ unzip 123010_fbo_ggs_Linux_x64_shiphome.zip
```

Create a silent installation response file, identifying the database version, GoldenGate Oracle Home, and database Oracle Home:

```
$ vi poplar.rsp
    INSTALL_OPTION=ORA12c
    SOFTWARE_LOCATION=/u01/app/oracle/product/ggpoplar
    START_MANAGER=true
    MANAGER_PORT=7809
    DATABASE_LOCATION=/u01/app/oracle/product/12.2.0/dbhome_1
    INVENTORY_LOCATION=/u01/app/oraInventory
    UNIX_GROUP_NAME=oinstall
```

Navigate to the directory containing the installer and invoke it in 'silent mode':

```
$ fbo_ggs_Linux_x64_shiphome/Disk1/runInstaller -silent \
  -nowait -responseFile ~/poplar.rsp
```

Patch GoldenGate

Optionally patch GoldenGate (using the opatch utility), after checking the Oracle Support notes (support.oracle.com) below for required or recommended patches:

- Master Note for Oracle GoldenGate Core Product Patch Sets (1645495.1);
- Recommended GoldenGate Patches (2193391.1).

Check and Prepare Source Database

GoldenGate captures database changes on the source database from database redo information. The source database must:

- be in archive log mode (so that no change information is lost during a significant outage);
- have minimal supplemental logging enabled (to pick up information on features such as row-chaining);
- have force logging enabled (to ensure details of all database changes appear in the redo stream).

```
SQL> select name, log_mode, supplemental_log_data_min, force_logging
     from v$database;

     NAME      LOG_MODE     SUPPLEME FOR
     --------- ------------ -------- ---
     SKYLARK   NOARCHIVELOG NO       NO

SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
```

```
SQL> alter database add supplemental log data;
SQL> alter database force logging;
SQL> alter system switch logfile;
SQL> select name, log_mode, supplemental_log_data_min, force_logging
     from v$database;

     NAME      LOG_MODE     SUPPLEME FOR
     --------- ------------ -------- ---
     SKYLARK   ARCHIVELOG   YES      YES
```

In Oracle database 11.2.0.4 and upwards, the parameter enable_goldengate_replication must be set to true to enable services, provided by the Oracle database, which are required by GoldenGate on the source database (e.g. supplemental logging required to support logical replication of some data types and operations, transparent data encryption handling):

```
SQL> alter system set enable_goldengate_replication=true scope=both;
```

Check and Prepare Target Database

The target database must be in archive log mode.

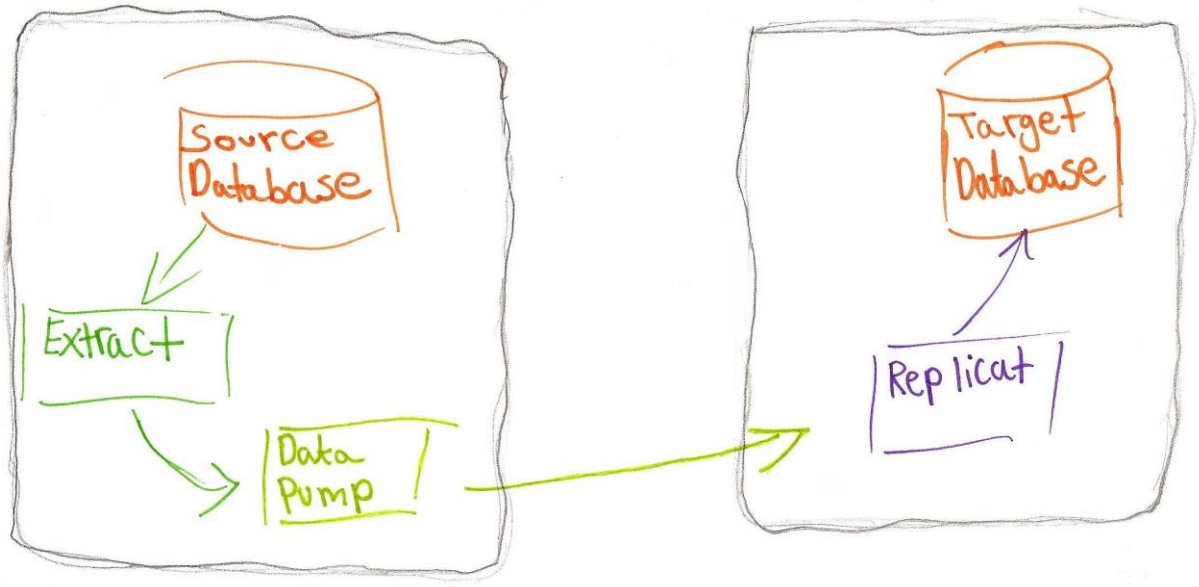
```
SQL> select name, log_mode from v$database;
     NAME      LOG_MODE
     --------- ------------
     PENGUIN   NOARCHIVELOG
SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
SQL> select name, log_mode from v$database;
     NAME      LOG_MODE
     --------- ------------
     PENGUIN   ARCHIVELOG
```

In Oracle database 11.2.0.4 and upwards, the parameter enable_goldengate_replication must be set to true to enable services, provided by the Oracle database, which are required by GoldenGate on the target database (e.g. trigger suppression, bypassing of referential integrity checking):

```
SQL> alter system set enable_goldengate_replication=true scope=both;
```

GoldenGate Architecture

Before configuring GoldenGate, it is necessary to understand its components. The diagram below shows a high-level overview of GoldenGate's architecture:



The Extract process captures database change information from the source database's redo data. It writes this information, in a proprietary format, to Trail Files.

The Data Pump process (not to be confused with Oracle Database Data Pump Import and Export) transfers the contents of these Trail Files to the target server (where they are known as Remote Trail Files).

The Replicat process reads the contents of these Trail Files and applies the database changes to the target database.

GoldenGate Extract and Replicat can run in one of two modes:

- Classic: where Extract reads the change data from the source database's transaction logs (i.e. redo data for an Oracle database) and where Replicat applies the changes via SQL;
- Integrated (which can only be used with Oracle databases): where Extract connects directly to the source database and obtains the database change information from the database's logminer process and where Replicat connects directly to the target database and applies the changes via the database's inbound server process.

As both methods communicate using identically formatted Trail Files, it is possible to mix Classic and Integrated Extract and Replicat, e.g. Classic Extract on the source database with Integrated Replicat on the target database.

Integrated Extract can be used with Oracle Database 11.2.0.3 onwards and Integrated Replicat with 11.2.0.4 onwards. Integrated Extract and Replicat leverage database code which was originally part of the now-deprecated Oracle Streams product.

Integrated Extract and Integrated Replicat will be configured in this document.

Configure Integrated Extract

Create a database user and tablespace for the Extract process:

```
SQL> create tablespace ggate datafile
     '/u01/app/oracle/oradata/SKYLARK/ggate01.dbf'
     size 20m autoextend on;
SQL> create user ggate identified by ggate default tablespace ggate;
```

Grant appropriate roles and permissions to the GoldenGate database user. These vary depending on the database version and the Extract mode. For Integrated Extract from Oracle Database 11.2.0.4 grant the following:

```
SQL> grant create session, connect, resource, alter any table,
     alter system, select any dictionary to ggate;
```

For Integrated Extract execute the command below to grant further required privileges:

```
SQL> exec dbms_goldengate_auth.grant_admin_privilege('ggate','capture');
```

Launch GGSCI (GoldenGate Software Command Interpreter) to complete the Extract configuration:

```
$ cd /u01/app/oracle/product/ggsycamore
$ ./ggsci
GGSCI (sycamore1) 1> dblogin userid ggate
```

Configure supplemental logging on the source database for tables or whole schemas which are to be replicated. The main purpose of this is the inclusion in the redo data of extra information (usually primary or unique key data for rows which have changed or been deleted) so that the same row can be identified on the target database. In this document, it is configured for the whole of the scott schema:

```
GGSCI (sycamore1) 2> add schematrandata scott
  2017-10-10 12:19:44  INFO    OGG-01788  SCHEMATRANDATA has been added on schema "scott".
  2017-10-10 12:19:44  INFO    OGG-01976  SCHEMATRANDATA for scheduling columns has been
                                          added on schema "scott".
```

Define an extract process to capture all changes from the scott schema. First create a parameter file for this extract process. Processes can have names of up to eight characters; ours is named 'exscott':

```
GGSCI (sycamore1) 3> edit params exscott
```

Within the parameter file define:

- the name of the extract;
- database connection information;
- location and name for Trail Files to be created by the extract. Ours will be stored in the dirdat directory below the GoldenGate Home. The name defined for Trail Files is a two-character prefix; ours is 'aa', to which GoldenGate will automatically append 6 digits;
- objects for which changes are to be extracted. In our case this is the whole scott schema.

```
EXTRACT exscott
USERID ggate, PASSWORD ggate
EXTTRAIL ./dirdat/aa
TABLE scott.*;
```

Use the 'add extract' command to create the extract, define it as an Integrated Extract and capture database changes from the current time onwards:

```
GGSCI (sycamore1) 4> add extract exscott, integrated tranlog, begin now
```

Use the 'add exttrail' command to create the local Trail Files and associate them with the exscott Extract:

```
GGSCI (sycamore1) 5> add exttrail ./dirdat/aa, extract exscott
```

Register the extract with the database, which allows Integrated Extract to communicate with the database's logminer process:

```
GGSCI (sycamore1) 6> register extract exscott database
```

Start the extract:

```
GGSCI (sycamore1) 7> start extract exscott
```

Use the 'info all' command to show the status of GoldenGate processes:

```
GGSCI (sycamore1) 8> info all
    Program     Status      Group       Lag at Chkpt  Time Since Chkpt
    MANAGER     RUNNING
    EXTRACT     RUNNING     EXSCOTT     00:00:00      00:01:25
```

Configure Data Pump

Define a Data Pump process to copy all local Trail Files to the target server. First create a parameter file for this Data Pump process. Ours is named 'dpscott':

```
GGSCI (sycamore1) 9> edit params dpscott
```

Within the parameter file define:

- the name of the Data Pump process (which is also, confusingly, defined as an 'EXTRACT');
- whether any data manipulation, transformation or filtering is to take place. In our case we define 'PASSTHRU' to state that the database changes will be passed from source to target unaltered (to allow this table definitions on source and target must be identical);
- the target server and the port on the target server on which GoldenGate will be listening;
- location and name for remote Trail Files. Ours will be stored in the dirdat directory below the GoldenGate Home. The name defined for Trail Files is a two-character prefix; ours is 'bb', to which GoldenGate will automatically append 6 digits;
- database objects for which changes are to be sent to the target server. In our case this is the whole scott schema.

```
EXTRACT dpscott
PASSTHRU
RMTHOST poplar1.localdomain, mgrport 7809
RMTTRAIL ./dirdat/bb
TABLE scott.*;
```

Use the 'add extract' command to create the Data Pump process, including the location and prefix for its local Trail Files:

```
GGSCI (sycamore1) 10> add extract dpscott, exttrailsource ./dirdat/aa
```

Use the 'add rmttrail' command to create the remote Trail Files and associate them with the dpscott Data Pump process:

```
GGSCI (sycamore1) 11> add rmttrail ./dirdat/bb, extract dpscott
```

Start the Data Pump process:

```
GGSCI (sycamore1) 12> start extract dpscott
```

Use the 'info all' command to show the status of GoldenGate processes:

```
GGSCI (sycamore1) 13> info all
    Program     Status      Group       Lag at Chkpt  Time Since Chkpt
    MANAGER     RUNNING
    EXTRACT     RUNNING     DPSCOTT     00:00:00      00:03:49
    EXTRACT     RUNNING     EXSCOTT     00:00:05      00:00:03
```

On the target server, a GoldenGate background process called the Collector (not shown on the architecture diagram) receives the data sent by the source server's Data Pump process and creates the remote Trail Files.

Instantiate the Target Database

Data changes are now being captured from the source database and are being transferred to the target server. However, the target database does not yet have a full copy of the schema(s) involved in the replication, to apply these changes to.

GoldenGate has an initial load facility available to carry out this 'instantiation' of the target database. It can copy data from and to differing database engines (e.g. Oracle to SQL Server, SQL Server to DB2).

Alternatively, if the source and target databases are both Oracle, then standard Oracle database methods can be used:

- Oracle Database Data Pump, i.e. expdp and impdp;
- RMAN (Recovery Manager), e.g. conventional backup and recovery, or duplication from backup or active database;
- Transportable Tablespaces.

We will use Oracle Database Data Pump.

There is no need to stop database updates to the source database during the Data Pump export; this allows applications accessing the source database to remain available. To achieve this, the export must be performed as of a known system change number.

In the source database (skylark) determine the current system change number:

```
SQL> select dbms_flashback.get_system_change_number from dual;

     GET_SYSTEM_CHANGE_NUMBER
     ------------------------
                       453114
```

On the source database server (sycamore1) create a Data Pump export parameter file, defining the schema to export and the system change number:

```
$ vi scott_exp.par

     dumpfile=scott.dmp
     logfile=scott_exp.log
     directory=exports
     schemas=scott
     flashback_scn=453114
```

Perform the Data Pump export:

```
$ expdp system parfile=scott_exp.par
```

Transfer the export dump file to the target server (poplar1):

```
$ scp scott.dmp poplar1:
```

In the target database, it is usually necessary to create a tablespace(s) for the schema(s) which are about to be created. In our example, we are using the scott schema, which uses the already-existing users tablespace.

On the target database server create a Data Pump import parameter file:

```
$ vi scott_imp.par

     dumpfile=scott.dmp
     logfile=scott_imp.log
     directory=exports
```

Perform the Data Pump import:

```
$ impdp system parfile=scott_imp.par
```

Configure Integrated Replicat

Create a database user and tablespace for the Replicat process:

```
SQL> create tablespace ggate datafile
     '/u01/app/oracle/oradata/PENGUIN/ggate01.dbf'
     size 20m autoextend on;
SQL> create user ggate identified by ggate default tablespace ggate;
```

Grant appropriate roles and permissions to the GoldenGate database user. These vary depending on the database version and the Replicat mode. For Integrated Replicat from Oracle Database 11.2.0.4 grant the following:

```
SQL> grant create session, connect, resource, dba to ggate;
```

For Integrated Replicat execute the command below to grant further required privileges:

```
SQL> exec dbms_goldengate_auth.grant_admin_privilege('ggate','apply');
```

Launch GGSCI to complete the Replicat configuration:

```
$ cd /u01/app/oracle/product/ggpoplar
$ ./ggsci
GGSCI (poplar1) 1> dblogin userid ggate
```

Define a Replicat process to apply all changes to the scott schema in the target database. First edit a parameter file for the Replicat process. Ours is named 'rpscott':

```
GGSCI (poplar1) 2> edit params rpscott
```

Within the parameter file:

- define the name of the replicat;
- database connection information;
- inform it, using the ASSUMETARGETDEFS parameter, that object definitions on source and target databases are identical;
- define the mapping of source database schema name to target database schema name. In our case these are identical, i.e. scott.

```
REPLICAT rpscott
USERID ggate, PASSWORD ggate
ASSUMETARGETDEFS
MAP scott.*, TARGET scott.*;
```

Use the 'add replicat' command to create the replicat, define it as an Integrated Replicat and give the location and prefix for the Trail Files from which it will read:

```
GGSCI (poplar1) 3> add replicat rpscott integrated, exttrail ./dirdat/bb
```

Start the Replicat, instructing it to only apply database changes which occurred after the system change number (which is called a CSN, Commit Sequence Number, in GoldenGate) at which the Data Pump export was performed:

```
GGSCI (poplar1) 5> start replicat rpscott, aftercsn 453114

GGSCI (poplar1) 6> info all
     Program     Status      Group       Lag at Chkpt  Time Since Chkpt
     MANAGER     RUNNING
     REPLICAT    RUNNING     RPSCOTT     00:00:07      00:00:09
```

GoldenGate replication is now configured and running between the source and target databases.

Monitor GoldenGate Status

To see the status of the Extract process:

```
GGSCI (sycamore1) 1> info extract exscott
     EXTRACT    EXSCOTT   Last Started 2017-10-10 12:23   Status RUNNING
     Checkpoint Lag       00:00:03 (updated 00:00:03 ago)
     Process ID           19367
     Log Read Checkpoint  Oracle Integrated Redo Logs
                          2017-10-10 15:31:23
                          SCN 0.461602 (461602)
```

To see the status of the Extract process in more detail:

```
GGSCI (sycamore1) 2> info extract exscott, detail
```

To see the status of the Data Pump process:

```
GGSCI (sycamore1) 3> info extract dpscott
     EXTRACT    DPSCOTT   Last Started 2017-10-10 12:43   Status RUNNING
     Checkpoint Lag       00:00:00 (updated 00:00:05 ago)
     Process ID           20520
     Log Read Checkpoint  File
/u01/app/oracle/product/ggsycamore/dirdat/aa000000000
                          2017-10-10 15:36:50.000000  RBA 849812
```

To see the status of the Data Pump process in more detail:

```
GGSCI (sycamore1) 4> info extract dpscott, detail
```

To see the status of the local and remote Trail Files:

```
GGSCI (sycamore1) 5> info exttrail *
           Extract Trail: ./dirdat/bb
             Seqno Length: 9
        Flip Seqno Length: no
                  Extract: DPSCOTT
                    Seqno: 0
                      RBA: 871868
                File Size: 500M
            Extract Trail: ./dirdat/aa
             Seqno Length: 9
        Flip Seqno Length: no
                  Extract: EXSCOTT
                    Seqno: 0
                      RBA: 872230
                File Size: 500M
```

To see statistics for the Extract process, i.e. numbers of rows inserted, updated etc.:

```
GGSCI (sycamore1) 6> stats extract exscott
     Sending STATS request to EXTRACT EXSCOTT ...
     Start of Statistics at 2017-10-10 15:43:53.
     Output to ./dirdat/aa:
     Extracting from SCOTT.PRODUCT to SCOTT.PRODUCT:
     *** Total statistics since 2017-10-10 12:23:57 ***
             Total inserts                                       0.00
             Total updates                                    3997.00
             Total deletes                                       0.00
             Total discards                                      0.00
             Total operations                                 3997.00
     *** Daily statistics since 2017-10-10 12:23:57 ***
             Total inserts                                       0.00
             Total updates                                    3997.00
             Total deletes                                       0.00
             Total discards                                      0.00
             Total operations                                 3997.00
     *** Hourly statistics since 2017-10-10 15:00:00 ***
             Total inserts                                       0.00
             Total updates                                     877.00
             Total deletes                                       0.00
             Total discards                                      0.00
             Total operations                                  877.00
     *** Latest statistics since 2017-10-10 12:23:57 ***
             Total inserts                                       0.00
             Total updates                                    3997.00
             Total deletes                                       0.00
             Total discards                                      0.00
             Total operations                                 3997.00
     End of Statistics.
```

To see the status of the Replicat process:

```
GGSCI (poplar1) 1> info replicat rpscott
     REPLICAT   RPSCOTT   Last Started 2017-10-10 15:23   Status RUNNING
     INTEGRATED
     Checkpoint Lag       00:00:09 (updated 00:00:10 ago)
     Process ID           31073
     Log Read Checkpoint  File
/u01/app/oracle/product/ggpoplar/dirdat/bb000000000
                          2017-10-10 16:19:11.680713  RBA 1041343
```

To see the status of the Replicat process in more detail:

```
GGSCI (poplar1) 2> info replicat rpscott, detail
```

To see by how much changes in the target database are lagging behind changes in the source database:

```
GGSCI (poplar1) 3> lag replicat rpscott
     Sending GETLAG request to REPLICAT RPSCOTT ...
     Last record lag 4 seconds.
     Low watermark lag: 8.
     High watermark lag: 5.
     Low watermark position: 468156.
     High watermark position: 468158.
     At EOF, no more records to process
```

There is a log file in the GoldenGate Home for all GoldenGate processes running on that server, e.g. /u01/app/oracle/product/ggsycamore/ggserr.log

For Integrated Extract and Replicat there are database data dictionary views, which show the status of the processes. These are dba_capture for Integrated Extract and dba_apply for Integrated Replicat:

```
SQL> select * from dba_capture;
SQL> select * from dba_apply;
```

Oracle provide health check scripts for Integrated Extract and Replicat, which can be found attached to Oracle Support note 1448324.1 (support.oracle.com). Download the version(s) of the script appropriate to the database version(s) of the source and target databases.

Logical Replication

Data replication products perform either physical replication or logical replication.

Physical replication products (e.g. Oracle Data Guard or storage replication provided by storage vendors) replicate database blocks from source to target database. This means that the source and target operating systems, database software, database character sets and database structures must be identical.

Logical replication products (e.g. GoldenGate, Dbvisit Replicate, Quest Shareplex) replicate database changes from source to target databases via SQL. They examine each data change on the source database and reverse-engineer SQL statements to apply the same changes to the target database.

This allows logical replication products to:

- replicate between database versions, e.g. Oracle 11.2 to Oracle 12.2;
- replicate between database types, e.g. SQL Server to Oracle;
- replicate between operating system, e.g. Windows to Linux;
- replicate between databases with differing character sets;
- replicate a subset of database objects;
- replicate a subset of rows;
- transform changes applied on the source database into alternative changes to be applied on the target database.

However, this flexibility comes with a price. There are important considerations associated with logical replication products:

- some data types may not be supported;
- it must be possible to uniquely identify a database row, so that, for a change applied to a row on the source database, the same row can be found on the target database. This is typically achieved with primary or unique keys; alternatively, it must be possible to identify a row using a defined combination of columns;
- set based operations become row-by-row operations, e.g. one update statement on the source database which updated 1,000 rows will be reverse-engineered by a logical replication product into 1,000 separate update statements.

The following articles by Arjen Visser of Dbvisit describe these considerations associated with logical replication:

- blog.dbvisit.com/3-fundamentals-of-oracle-replication-part-1-of-3
- blog.dbvisit.com/the-3-fundamentals-of-oracle-replication-part-2-of-3
- blog.dbvisit.com/httpblog-dbvisit-com3-fundamentals-of-oracle-replication-part-of-3/

Below is an example which shows set based operations becoming row-by-row.

Execute a multiple row update statement on the source database:

```
SQL> update scott.emp set hiredate = sysdate;
14 rows updated.
```

The SQL statement was executed once and used a full table scan:

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |      |       |       |     3 (100)|          |
|   1 |  UPDATE            | EMP  |       |       |            |          |
|   2 |   TABLE ACCESS FULL| EMP  |    14 |   168 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

GoldenGate Replicat applies the same data change to the target database by executing the statement below:

```
UPDATE "SCOTT"."EMP" SET "HIREDATE"=:1 WHERE "EMPNO"=:2
```

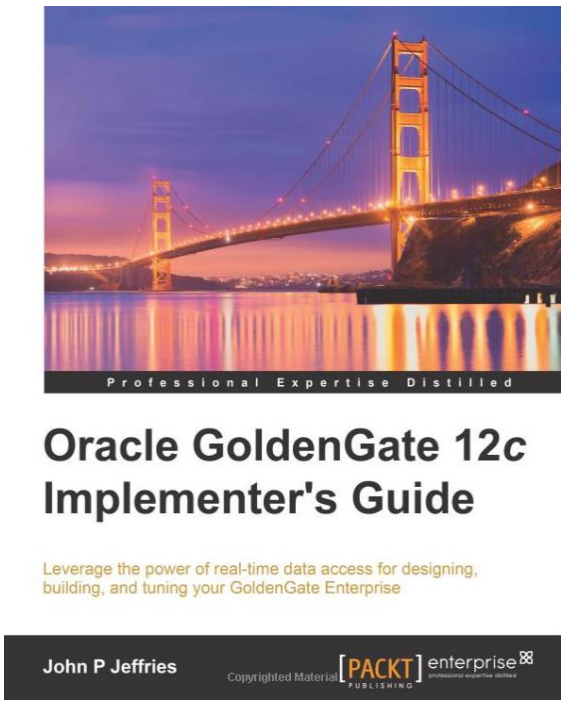
The SQL Statement was executed fourteen times and used index access:

```
-------------------------------------------------------------------
| Id  | Operation          | Name   | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------
|   0 | UPDATE STATEMENT   |        |       |       |     1 (100)|
|   1 |  UPDATE            | EMP    |       |       |            |
|*  2 |   INDEX UNIQUE SCAN| PK_EMP |     1 |    12 |     0   (0)|
-------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("EMPNO"=:2)
```

The fact that set based operations must become row-by-row operations on a target database can lead to a target database lagging a long way behind its source database, after large multi-row operations on the source database.

Recommended Reading

Oracle Goldengate 12c Implementer's Guide by John P Jeffries



Conclusion

This document has covered

- GoldenGate installation;
- configuration of database replication;
- Goldengate monitoring;
- key principles of logical replication.

I hope this has whetted your appetite and inspired you to investigate Oracle GoldenGate further.

Contact address:

Patrick Hurley
Accenture Enkitec Group
Warwick Technology Park
Gallows Hill
Warwick
CV34 6DW

Phone: +44(0)7769-243584
Email patrick.hurley@accenture.com
Internet: patrickhurley.wordpress.com