How to Handle Tables Without Primary Keys or Unique Indexes With Oracle GoldenGate (Doc ID 1271578.1)

APPLIES TO:

Oracle GoldenGate
Information in this document applies to any platform.

ABSTRACT

*

HISTORY

*

DETAILS

*

SUMMARY

Enter the Main Content

Deploying Oracle GoldenGate to Achieve Operational Reporting for Oracle E-Business Suite

This knowledge document describes a best practice method for handling tables without primary keys or unique indexes when using Oracle GoldenGate to replicate transactional data between Oracle databases.

There is a change log at the end of this document.

In This Document

This document is divided into the following sections:

Section 1: Overview

1.1 Solution Summary

In order to maintain data integrity when replicating transactional data, Oracle GoldenGate will use primary key columns or unique index columns to uniquely identify a row when issuing update or delete statements against the target database. If no primary keys or unique indexes exist on the table being replicated, Oracle GoldenGate will use all columns to uniquely identify a row.

It is perfectly acceptable to use all columns to uniquely identify a row under the following conditions:

- A logical key column cannot be defined for the table using the KEYCOLS parameter.
- No duplicate rows exist in the table

- Table contains a small number of rows, so full table lookups on the target database are minimal
- Table DML activity is very low, so "all column" table supplemental log groups do not negatively impact the source database redo logs

If the table being replicated does not meet all of the conditions listed above, it is recommended to add a column to the table with a SYS_GUID default value to uniquely identify the row. The next section describes the detailed steps on how to configure a table without a primary key or unique index in order to uniquely identify each row.

1.2 Required Software Components

Software Component	Minimum Version
Oracle Database	9.2 or greater
Oracle GoldenGate	10.4 or greater

Section 2: Configuring Tables without PKs or UIs

2.1 Configure Table(s) in Source Database

Before instantiating the target database from a copy of source, perform the following steps to the table(s) without primary keys or unique indexes.

Step 1 - Add Column to Table

Issue the following command to add the column to the table.

Replace <table_name> with the table name being modified.

```
alter table <table_name> add OGG_KEY_ID raw(16);
```

Step 2 - Modify Column Data Default to Use SYS_GUID Values

Issue the following command to modify the OGG_KEY_ID default value to use SYS_GUID values. SYS_GUID values are unique values generated from an internal Oracle algorithm.

Immediately after modifying the OGG_KEY_ID column, newly inserted rows will automatically populate the OGG_KEY_ID column with SYS_GUID values.

Replace <table_name> with the table name being modified.

```
alter table <table_name) modify OGG_KEY_ID default sys_guid();
```

Note: DO NOT combine steps 1 and 2 into 1 SQL statement. Doing so will cause Oracle to populate the OGG_KEY_ID column with a full table lock. The table could be locked for a significant amount of time based on the number of existing rows in the table.

Step 3 - Backfill Existing Table Rows with SYS_GUID Values

Now that newly inserted rows are being handled, the OGG_KEY_ID column still needs to be populated for existing table rows. The following SQL script will backfill the existing table rows with unique SYS_GUID values.

Replace <table_name> with the table name being updated.

```
DECLARE cursor C1 is select ROWID from
<table_name>
where OGG_KEY_ID is null;
finished number:=0; commit_cnt number:=0; err_msg varchar2(150);
snapshot_too_old exception; pragma exception_init(snapshot_too_old, -1555);
old_size number:=0; current_size number:=0;
BEGIN
while (finished=0) loop
finished:=1;
BEGIN
for C1REC in C1 LOOP
update <table_name>
set OGG_KEY_ID = sys_guid()
where ROWID = C1REC.ROWID;
commit_cnt:= commit_cnt + 1;
IF (commit_cnt = 10000) then
commit;
            commit_cnt:=0;
END IF;
END LOOP;
EXCEPTION
when snapshot_too_old then
            finished:=0;
when others then
            rollback;
            err_msg:=substr(sqlerrm,1,150);
            raise_application_error(-20555, err_msg);
END;
END LOOP;
IF(commit_cnt > 0) then
commit;
END IF;
END;

/
```

Note: This process could take a significant amount of time to complete based on the number of existing rows in the table. The script will not perform a full table lock and will only lock each row exclusively as it updates it.

Step 4 - Create Table Supplemental Log Group in Source Database

Issue the following commands in GGSCI to create the table supplemental log group using the OGG_KEY_ID column. This forces Oracle to always write the OGG_KEY_ID value to the online redo logs.

Replace <username> with source database username.

Replace <password> with source database user's password.

Replace <owner> with table schema owner.

Replace <table_name> with the table name being modified.

```
GGSCI> dblogin userid <username>, password <password>

GGSCI> add trandata <owner>.<table_name>, COLS (OGG_KEY_ID), nokey
```

2.2 Configure Table in Target Database

Step 1 - Create Unique Index on Target Table

After instantiating the target database, issue the following command to add a unique index to the table. The unique index prevents full table scans when applying update and delete statements to the target database.

Replace <table_name> with the table name the unique index is on.

Replace <tablespace_name> with tablespace name to store unique index.

```
create unique index OGG_<table_name>_UI on <table_name> (OGG_KEY_ID) logging online tablespace
<tablespace_name>
```

Note: This process could take a significant amount of time to complete based on the number of existing rows in the table.

2.3 Oracle GoldenGate Configuration

Step 1 - Specify OGG_KEY_ID for Table Key in Extract Parameter File

Use the KEYCOLS parameter in the Extract parameter file to define OGG_KEY_ID as the unique column for the table.

Replace <table_name> with the table name the unique index is on.

```
TABLE <table_name>, KEYCOLS (OGG_KEY_ID);
```

Section 3: References

For further information about using Oracle GoldeGate, refer to the following documents available on edelivery:

- Oracle GoldenGate Installation and Setup Guides
- Oracle GoldenGate Administration Guide
  - Introduces Oracle GoldenGate components and explains how to plan for, configure, and implement Oracle GoldenGate.
- Oracle GoldenGate Reference Guide
  - Provides detailed information about Oracle GoldenGate parameters, commands, and functions.
- Oracle GoldenGate Troubleshooting and Performance Tuning Guide
  - Provides suggestions for improving the performance of Oracle GoldenGate in different situations, and provides solutions to common problems.

Appendix A: Sample Table Configuration

```
Add Column to Table
```

```
SQL> SELECT * FROM NL_WEST;

TEAM_NAME         LOCATION
---------------   --------------------
PADRES            SAN DIEGO
ROCKIES           COLORADO
DODGERS           LOS ANGELES
DIAMONDBACKS      ARIZONA

SQL> alter table NL_WEST add OGG_KEY_ID raw(16);

Table altered.

SQL> select * from NL_WEST;

TEAM_NAME         LOCATION            OGG_KEY_ID
---------------   ------------------  ----------------
PADRES            SAN DIEGO
ROCKIES           COLORADO
DODGERS           LOS ANGELES
DIAMONDBACKS      ARIZONA
```

Modify Column Data Default to Use SYS_GUID() Values

```
SQL> alter table NL_WEST modify OGG_KEY_ID default sys_guid();

Table altered.

SQL> select * from NL_WEST;

TEAM_NAME         LOCATION            OGG_KEY_ID
---------------   ------------------  ----------------
PADRES            SAN DIEGO
ROCKIES           COLORADO
DODGERS           LOS ANGELES
DIAMONDBACKS      ARIZONA

SQL> INSERT INTO NL_WEST (TEAM_NAME,LOCATION) VALUES ('GIANTS','SAN FRANCISCO');

1 row created.

SQL> COMMIT;

Commit complete.

SQL> select * from NL_WEST;

TEAM_NAME         LOCATION          OGG_KEY_ID
---------------   ----------------  ----------------
PADRES            SAN DIEGO
ROCKIES           COLORADO
DODGERS           LOS ANGELES
DIAMONDBACKS      ARIZONA
GIANTS            SAN FRANCISCO     95AB2831E724991FE040C8C86E0162D0
```

Backfill Existing Table Rows with SYS_GUID() Values

```
SQL> DECLARE cursor C1 is select ROWID from
NL_WEST
where OGG_KEY_ID is null;
2 3 4 finished number:=0; commit_cnt number:=0; err_msg varchar2(150);
5 snapshot_too_old exception; pragma exception_init(snapshot_too_old, -1555);
6 old_size number:=0; current_size number:=0;
7 BEGIN
8 while (finished=0) loop
9 finished:=1;
10 BEGIN
11 for C1REC in C1 LOOP
```

```
12 update NL_WEST
13 set OGG_KEY_ID = sys_guid()
14 where ROWID = C1REC.ROWID;
15 commit_cnt:= commit_cnt + 1;
16 IF (commit_cnt = 10000) then
17 commit;
18 commit_cnt:=0;
19 END IF;
20 END LOOP;
21 EXCEPTION
22 when snapshot_too_old then
23 finished:=0;
24 when others then
25 rollback;
26 err_msg:=substr(sqlerrm,1,150);
27 raise_application_error(-20555, err_msg);
28 END;
END LOOP;
29 30 IF(commit_cnt > 0) then
31 commit;
32 END IF;
33 END;
34 /

PL/SQL procedure successfully completed.

SQL> select * from NL_WEST;

TEAM_NAME         LOCATION          OGG_KEY_ID
----------------  ----------------  ----------------
PADRES            SAN DIEGO         95AB2831E725991FE040C8C86E0162D0
ROCKIES           COLORADO          95AB2831E726991FE040C8C86E0162D0
DODGERS           LOS ANGELES       95AB2831E727991FE040C8C86E0162D0
DIAMONDBACKS      ARIZONA           95AB2831E728991FE040C8C86E0162D0
GIANTS            SAN FRANCISCO     95AB2831E724991FE040C8C86E0162D0
```

Create Table Supplemental Log Group in Source Database

```
GGSCI> dblogin userid OGG, password OGG

GGSCI> add trandata MLB.NL_WEST, COLS (OGG_KEY_ID), nokey
```

Create Unique Index on Target Table

```
SQL> create unique index GG_NL_WEST_UI on NL_WEST (OGG_KEY_ID) logging online;

Index created.
```

Change Log

Date	Description
December 3, 2010	• Initial creation.