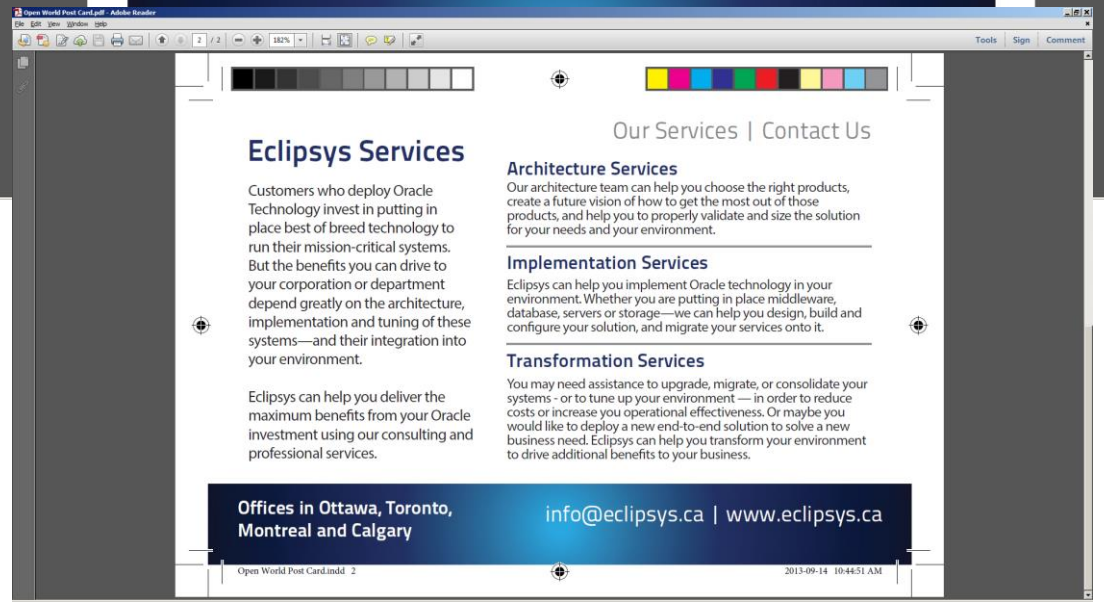
Become a Golden DBA for GoldenGate Environment

About Eclipsys

- Oracle Platinum Partner
- Market: **Canada**
- Services: (Most of Oracle stack)
  - Architecture
  - Implementation
  - Transformation
  - Managed Services
- Customers:
  - Public sector (60%)
  - Private/commercial sector (40%)

ORACLE Platinum Partner



Our Services | Contact Us

Eclipsys Services

Customers who deploy Oracle Technology invest in putting in place best of breed technology to run their mission-critical systems. But the benefits you can drive to your corporation or department depend greatly on the architecture, implementation and tuning of these systems—and their integration into your environment.

Eclipsys can help you deliver the maximum benefits from your Oracle investment using our consulting and professional services.

Architecture Services

Our architecture team can help you choose the right products, create a future vision of how to get the most out of those products, and help you to properly validate and size the solution for your needs and your environment.

Implementation Services

Eclipsys can help you implement Oracle technology in your environment. Whether you are putting in place middleware, database, servers or storage—we can help you design, build and configure your solution, and migrate your services onto it.

Transformation Services

You may need assistance to upgrade, migrate, or consolidate your systems - or to tune up your environment — in order to reduce costs or increase you operational effectiveness. Or maybe you would like to deploy a new end-to-end solution to solve a new business need. Eclipsys can help you transform your environment to drive additional benefits to your business.

Offices in Ottawa, Toronto, Montreal and Calgary

info@eclipsys.ca | www.eclipsys.ca

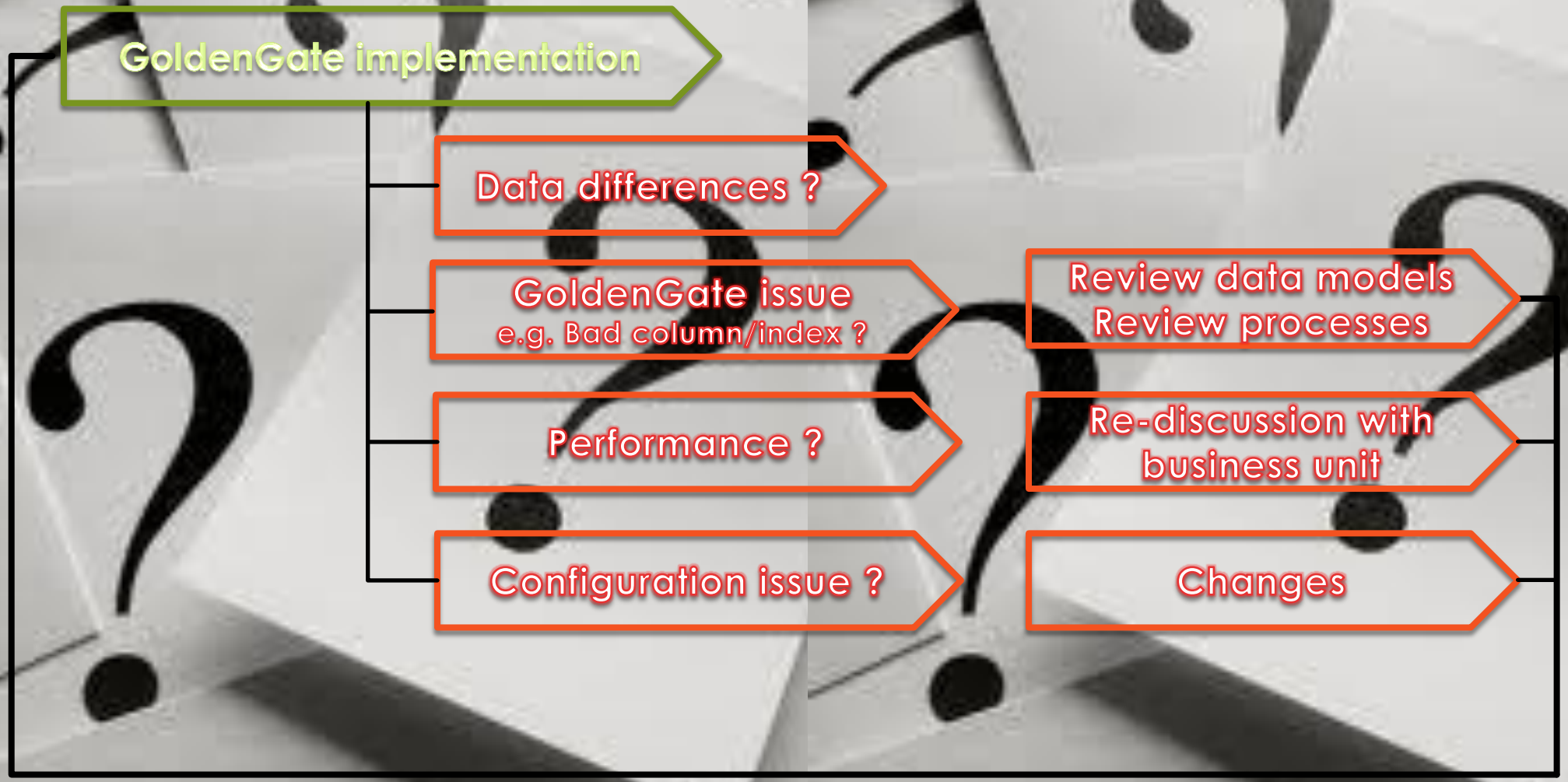
Open World Post Card.indd 2 2013-09-14 10:44:51 AM

About myself

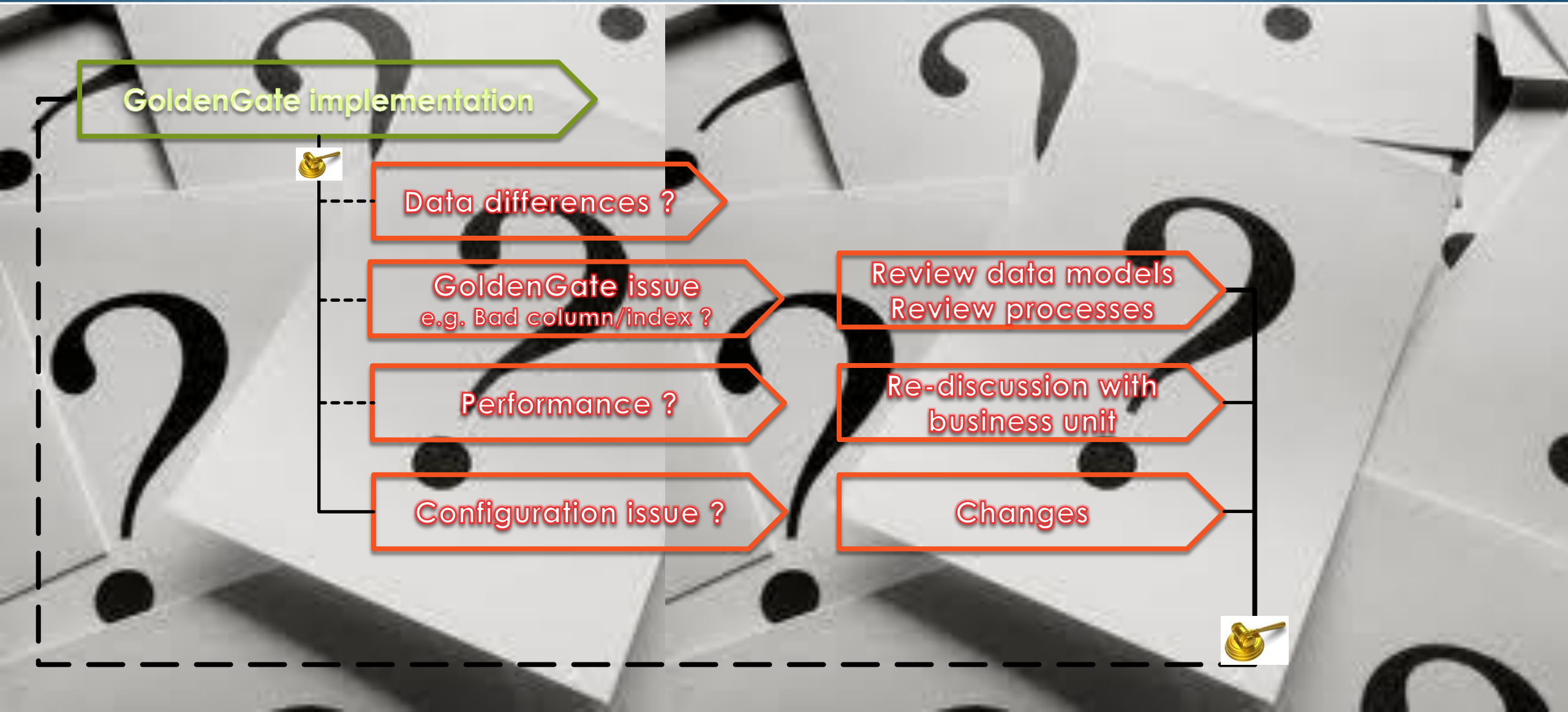
Shervin Sheidaei

- Practice lead on database and engineered systems at Eclipsys
- Oracle DBA since 2001
- Blogger
  - oradbatips.blogspot.com
- GoldenGate :
  - 6 nodes active-active GoldenGate implementation
  - GoldenGate on distributed Oracle Database Appliance
  - Speaker at conferences : OOW 2011, UKOUG 2012

Goal – Golden DBA



Goal – Golden DBA



Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- Replication between different structures
- What specific to monitor
- Additional quick tips
- Upcoming topics

Assumptions

- ✓ List of replicated tables are determined
- ✓ GoldenGate is installed
- ✓ All tests are done on GoldenGate 11.2 with Oracle 11g
- ✓ Replication is between Oracle databases (Source and Target)
- ✓ Test is done on GoldenGate classic capture.

Basic knowledge of GoldenGate

First step– Where to start

Oracle GoldenGate database Complete Database Profile check script for Oracle DB (All Schemas) (Doc ID 1298562.1)

```
------ Distinct Column Data Types and their Count in the Schema: JIRA

DATA_TYPE            TOTAL
---------------- ---------
LONG RAW               1
TIMESTAMP(6)           1
FLOAT                  2
NUMBER               333
CLOB                  16
CHAR                   1
DATE                  48
VARCHAR2             396

------ Tables With No Primary Key or Unique Index in the Schema: JIRA

------ Tables With CLOB, BLOB, LONG, NCLOB or LONG RAW Columns in the Schema: JIRA

TABLE_NAME                     COLUMN_NAME                    DATA_TYPE
------------------------------ ------------------------------ ----------
AO_563AEE_ACTIVITY_ENTITY      CONTENT                        CLOB
CHANGEITEM                     OLDVALUE                       CLOB
CHANGEITEM                     OLDSTRING                      CLOB
CHANGEITEM                     NEWVALUE                       CLOB
CHANGEITEM                     NEWSTRING                      CLOB
CUSTOMFIELDVALUE               TEXTVALUE                      CLOB
GADGETUSERPREFERENCE           USERPREFVALUE                  CLOB
JIRAACTION                     ACTIONBODY                     CLOB
JIRADRAFTWORKFLOWS             DESCRIPTOR                     CLOB
JIRAISSUE                      DESCRIPTION                    CLOB
JIRAISSUE                      ENVIRONMENT                    CLOB
JIRAWORKFLOWS                  DESCRIPTOR                     CLOB
PROPERTYDATA                   PROPERTYVALUE                  LONG
PROPERTYTEXT                   PROPERTYVALUE                  CLOB
SEARCHREQUEST                  REQCONTENT                     CLOB
USERHISTORYITEM                DATA                           CLOB
WORKLOG                        WORKLOGBODY                    CLOB

------ Types of Constraints on the Tables in the Schema: JIRA

CONSTRAINT_TYPE_DESC                 TOTAL
------------------------------ ---------
REFERENTIAL                           15
PRIMARY KEY                          140
CHECK                                182

------ Tables Defined with Triggers in the Schema: JIRA

TABLE_NAME                     TRIGGER_COUNT
------------------------------ -------------
AO_60DB71_SPRINT                           1
AO_60DB71_ISSUERANKING                     1
AO_563AEE_ACTIVITY_ENTITY                  1
AO_60DB71_ESTIMATESTATISTIC                1
AO_60DB71_SWIMLANE                         1
AO_60DB71_RAPIDVIEW                        1
AO_60DB71_RANKABLEOBJECT                   1
AO_60DB71_COLUMN                           1
AO_60DB71_CARDCOLOR                        1
AO_563AEE_MEDIA_LINK_ENTITY                1
AO_563AEE_ACTOR_ENTITY                     1
AO_60DB71_QUICKFILTER                      1
AO_60DB71_COLUMNSTATUS                     1
AO_563AEE_TARGET_ENTITY                    1
AO_60DB71_ISSUERANKINGLOG                  1
AO_60DB71_SUBQUERY                         1
AO_563AEE_OBJECT_ENTITY                    1
```

First step– Where to start

Oracle GoldenGate database Complete Database Profile check script for Oracle DB (All Schemas) (Doc ID 1298562.1)

- Object Types and their Count
- Column Data Types and their Count
- Special :IOT,cluster table, domain index
- Tables With No Primary Key or Unique Index
- Tables With No Primary Key or Unique Index >1M
- Types of Constraints on the Tables in the Schema
- Tables with Deferred constraints
- Tables with cascade deletes
- Tables with triggers

Limitation/Unsupported
GG version
DDL replication

Oracle GoldenGate Oracle Installation and Setup Guide

Further table analysis
Business rules/Application rules
Type of changes on these tables

Cross schema constraints
Cascade delete on target??
Architecture of replicat, split

Further analysis of trigger changes
Manual disabling of triggers
Managing triggers via GoldenGate??

First step– Where to start

Oracle GoldenGate database Complete Database Profile check script for Oracle DB (All Schemas) (Doc ID 1298562.1)

- **Status of supplemental logging**
- **Frequency of log switches**
- **Break down of log generation per hour**

→ Supplemental level
Estimated spec for disk/network
Options for compression

- **Tables With Nologging operation**

→ Further table analysis
Impact of force logging
Replication method

- **Compressed table**
- **Database type –RAC/RAC One/EE/SE**

→ GoldenGate capture method
GoldenGate version

- **Specific object : Sequence**

→ Further analysis
Architect replicat/capture

Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- Replication between different structures
- What specific to monitor
- Additional quick tips
- Upcoming topics

Parameter file and Table

Table with Primary key or unique key

Table with set of columns

Table with no unique rows

- **Each row is unique***
  - Name table in capture
  - Use wildcard when there are many tables
  - Sample for capture:
    - table repuser2.testpk
    - table repuser2.*

- **Application logic makes each row to be unique**
  - Use keycols for combination of columns which are unique
  - Sample :
    - table repuser2.testkeycol, keycol(col1,col2)

- **No unique row**
  - All rows are captured by default (small table/low DML activity)
  - Issue only for update/delete. No issue with insert only table
    - Manual copy
    - Move to keycol
    - Introduce a unique column (a column with SYS_GUID) **Doc ID 1271578.1**

Parameter file and Table

Table with PK/UI

Table with KEYCOL

Source

```
TABLE   repuser2.testkeycol, keycol(col1,col2);

TABLE  repuser2.*;
```

Target

```
MAP repuser2.*, TARGET repuser2.*;
```

Source

```
desc repuser2.testkeycol

col1   number

col2 varchar2(30)

Col3 varchar2(30)

insert into repuser2.testkeycol
values(10,1000,'Row10');

commit;
```

GG replication

Target

```
select * from repuser2.testkeycol where
col1=10;

COL1   COL2   COL3

--------------------------

10       1000    Row10

10       1000    Row10
```

Parameter file and Table

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

TABLEEXCLUDE repuser2.testkeycol;

TABLE repuser2.*;

Target

MAP repuser2.*, TARGET repuser2.*;

Table with PK/UI

Table with KEYCOL

Source

insert into repuser2.testkeycol values(30,3000,'Row30');

commit;

GG replication

Target

select * from repuser2.testkeycol where col1=30;

No rows

Parameter file and Table

Table with PK/UI

Table with KEYCOL

Map with PK/UI

Map with KEYCOL

Source

```
TABLE   repuser2.testkeycol, keycol(col1,col2);

TABLE  repuser2.*;
```

Target

```
MAP repuser2.testkeycol, TARGET
repuser2.testkeycol, KEYCOLS (col1,col2);

MAP repuser2.*, TARGET repuser2.*;
```

Source

```
insert into repuser2.testkeycol
values(40,4100,'Row41');

commit;
```

GG replication

Target

```
select * from repuser2.testkeycol where col1=40;

COL1   COL2   COL3
-----------------------------
40     4100   Row41
40     4100   Row41
40     4100   Row41
40     4100   Row41
```

Parameter file and Table

- Table with PK/UI
- Table with KEYCOL
- Map with PK/UI
- Map with KEYCOL
- Table excluded

Source

```
WILDCARDRESOLVE IMMEDIATE
--tables with no PK/UI but with specific  keycols
INCLUDE ./dirprm/key_cols_tabs.inc
WILDCARDRESOLVE DYNAMIC
-- Exclude keycol tables
INCLUDE ./dirprm/key_cols_tabs_exclusion.inc
-- Exclude tables which are excluded from replication.
INCLUDE ./dirprm/eliminated_tabs.inc
-- Replicat all other tables which have PK/UI.
INCLUDE ./dirprm/pk_ui_tabs.inc
```

Target

```
WILDCARDRESOLVE IMMEDIATE
-- map tables with no PK/UI but with specific  keycols
INCLUDE ./dirprm/key_cols_tabs.inc
WILDCARDRESOLVE DYNAMIC
-- Exclude keycol tables
INCLUDE ./dirprm/key_cols_tabs_exclusion.inc
-- Exclude tables which are not needed
INCLUDE ./dirprm/exclude_tabs.inc
-- Replicat all other tables which have PK/UI.
INCLUDE ./dirprm/pk_ui_tabs.inc
```

Pump (Optional)

```
-- Extract whatever master ext extracted.
INCLUDE ./dirprm/pk_ui_tabs.inc
```

Parameter file and Table

Table with PK/UI

Table with KEYCOL

Map with PK/UI

Map with KEYCOL

Table excluded

Source

key_cols_tabs.inc

table repuser2.testkeycol, keycol(col1,col2);

key_cols_tabs_exclusion.inc

TABLEEXCLUDE repuser2.testkeycol;

eliminated_tabs.inc

TABLEEXCLUDE repuser2.dummy_table;

pk_ui_tabs.inc

TABLE repuser2.*;

Target

key_cols_tabs.inc

MAP repuser2.testkeycol, TARGET repuser2.testkeycol, KEYCOLS (col1,col2);

key_cols_tabs_exclusion.inc

MAPEXCLUDE repuser2.testkeycol;

exclude_tabs.inc

MAPEXCLUDE repuser2.seq_*;

pk_ui_tabs.inc

TABLE repuser2.*;

Source

insert into repuser2.testkeycol values(60,600,'Row6');

commit;

GG replication

Target

select * from repuser2.testkeycol where col1=60;

```
COL1    COL2    COL3
-------------------------
60      600     Row6
```

First step– Where to start

Category tables to 3 types
- Excluded table
- Table with keycol
- Table with PK/UI

Use include file for each type

WILDCARDRESOLVE IMMEDIATE for keycol

Keycol tables to be excluded if wildcard is used

Consistency between include files in capture and replicat

Validate No duplicates among various process

Source

```
WILDCARDRESOLVE IMMEDIATE
--tables with no PK/UI but with specific  keycols
INCLUDE ./dirprm/key_cols_tabs.inc
WILDCARDRESOLVE DYNAMIC
-- Exclude keycol tables
INCLUDE ./dirprm/key_cols_tabs_exclusion.inc
-- Exclude tables which are excluded from replication.
INCLUDE ./dirprm/eliminated_tabs.inc
-- Replicat all other tables which have PK/UI.
INCLUDE ./dirprm/pk_ui_tabs.inc
```

Target

```
WILDCARDRESOLVE IMMEDIATE
-- map tables with no PK/UI but with specific  keycols
INCLUDE ./dirprm/key_cols_tabs.inc
WILDCARDRESOLVE DYNAMIC
-- Exclude keycol tables
INCLUDE ./dirprm/key_cols_tabs_exclusion.inc
-- Exclude tables which are not needed
INCLUDE ./dirprm/exclude_tabs.inc
-- Replicat all other tables which have PK/UI.
INCLUDE ./dirprm/pk_ui_tabs.inc
```

Use template for any implementation

Keycol tables

Table with set of columns (Keycol table)

- Set of columns that will uniquely identify the each row
- Currently no duplicate records exist
- Analyze/Examine batch operations such as batch delete

Table with no unique rows

- 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

Keycol tables

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

Keycol table

Source

desc repuser2.testkeycol

col1 number

col2 varchar2(30)

Col3 varchar2(30)

add trandata repuser2.testkeycol, nokey, cols (col1,col2)

select a.* from dba_log_group_columns a where table_name='TESTKEYCOL';

OWNER	LOG_GROUP_NAME	TABLE_NAME	COLUMN_NAME	POSITION	LOGGING_PROPERTY
REPUSER2	GGS_109979	TESTKEYCOL	COL1	1	LOG
REPUSER2	GGS_109979	TESTKEYCOL	COL2	2	LOG

Target

select * from repuser2.testkeycol;

No rows

Keycol tables

Keycol table

Source

```
TABLE   repuser2.testkeycol, keycol(col1,col2);
```

Target

```
MAP REPUSER2.TESTKEYCOL, TARGET
REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;
```

Source

```
insert into repuser2.testkeycol values(1,100,'Row1');
insert into repuser2.testkeycol values(2,200,'Row2');
insert into repuser2.testkeycol values(null,null,'Row3');
insert into repuser2.testkeycol values(null,null,'Row4');
insert into repuser2.testkeycol values(5,500,'Row5');
```

GG replication

Target

```
select * from repuser2.testkeycol

COL1   COL2   COL3
-------------------------------
1      100    Row1
2      200    Row2
              Row3
              Row4
5      500    Row5
```

Keycol tables

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

Keycol table

Source

select rowid,a.* from repuser2.testkeycol a

ROWID	COL1	COL2	COL3
AAAa2bAAEAAAIRVAAA	1	100	Row1
AAAa2bAAEAAAIRVAAB	2	200	Row2
AAAa2bAAEAAAIRVAAC			Row3
AAAa2bAAEAAAIRVAAD			Row4
AAAa2bAAEAAAIRVAAE	5	500	Row5

Updating 4th row

update repuser2.testkeycol set col1=4,col2=400 where rowid='AAAa2bAAEAAAIRVAAD';

Commit;

GG replication

Target

select * from repuser2.testkeycol

COL1	COL2	COL3
1	100	Row1
2	200	Row2
4	400	Row3
		Row4
5	500	Row5

Keycol tables

Source

```
TABLE   repuser2.testkeycol, keycol(col1,col2);
```

Target

```
MAP REPUSER2.TESTKEYCOL, TARGET
REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;
```

Fix ? Adding supplemental logging

Source

```
add trandata repuser2.testkeycol, nokey, cols
(col1,col2,col3)

select a.* from dba_log_group_columns a where
table_name='TESTKEYCOL';

OWNERLOG_GROUP_NAME    TABLE_NAME
        COLUMN_NAME        POSITION
        LOGGING_PROPERTY

REPUSER2        GGS_109979    TESTKEYCOL     COL1    1
        LOG

REPUSER2        GGS_109979    TESTKEYCOL     COL2    2
        LOG

REPUSER2        GGS_109979    TESTKEYCOL     COL3    3
        LOG
```

Keycol tables

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

Fix ? Adding supplemental logging

Source

update repuser2.testkeycol set col1=6,col2=600 where rowid='AAAa2bAAEAAAIRVAAD';

commit;

select rowid,a.* from repuser2.testkeycol a

ROWID	COL1	COL2	COL3
AAAa2bAAEAAAIRVAAA	1	100	Row1
AAAa2bAAEAAAIRVAAB	2	200	Row2
AAAa2bAAEAAAIRVAAC			Row3
AAAa2bAAEAAAIRVAAD	6	600	Row4
AAAa2bAAEAAAIRVAAE	5	500	Row5

GG replication

Target

select * from repuser2.testkeycol;

COL1	COL2	COL3
1	100	Row1
2	200	Row2
6	600	Row4
		Row4
5	500	Row5

UPDATE "REPUSER2"."TESTKEYCOL" SET "COL1" = :a2,"COL2" = :a3,"COL3" = :a4 WHERE "COL1" = :b0 AND "COL2" = :b1 AND ROWNUM = 1

Keycol tables

Source

update repuser2.testkeycol set col1=6,col2=600 where rowid='AAAa2bAAEAAAIRVAAD';

Fix ? Adding supplemental logging?
ANSWER: NO

Target

UPDATE "REPUSER2"."TESTKEYCOL" SET "COL1" = :a2,"COL2" = :a3,"COL3" = :a4 WHERE "COL1" = :b0 AND "COL2" = :b1 AND ROWNUM = 1

Where condition in replicat is built based on defined keycols in replicat parameter file

Keycol tables

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

What about delete ?

Source

insert into repuser2.testkeycol values(null,null,'Row6');

Commit;

select rowid,a.* from repuser2.testkeycol a;

ROWID	COL1	COL2	COL3
AAAa2bAAEAAAIRVAAA	1	100	Row1
AAAa2bAAEAAAIRVAAB	2	200	Row2
AAAa2bAAEAAAIRVAAC			Row3
AAAa2bAAEAAAIRVAAD	6	600	Row4
AAAa2bAAEAAAIRVAAE	5	500	Row5
AAAa2bAAEAAAIRXAAA			Row6

GG replication

Target

select * from repuser2.testkeycol

COL1	COL2	COL3
1	100	Row1
2	200	Row2
		Row3
6	600	Row4
5	500	Row5
		Row6

Keycol tables

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

What about delete ?

Source

select rowid,a.* from repuser2.testkeycol a;

ROWID	COL1	COL2	COL3
AAAa2bAAEAAAIRVAAA	1	100	Row1
AAAa2bAAEAAAIRVAAB	2	200	Row2
AAAa2bAAEAAAIRVAAC			Row3
AAAa2bAAEAAAIRVAAD	6	600	Row4
AAAa2bAAEAAAIRVAAE	5	500	Row5
AAAa2bAAEAAAIRXAAA			Row6 --> delete this row....

delete from repuser2.testkeycol where rowid='AAAa2bAAEAAAIRXAAA';

commit;

GG replication

Target

select * from repuser2.testkeycol

COL1	COL2	COL3
1	100	Row1
2	200	Row2
6	600	Row4
5	500	Row5
		Row6 ---> row3 was deleted !!!

DELETE FROM "REPUSER2"."TESTKEYCOL" WHERE "COL1" is NULL AND "COL2" is NULL AND ROWNUM = 1

Keycol tables

Source

```
TABLE   repuser2.testkeycol, keycol(col1,col2);
```

Target

```
MAP REPUSER2.TESTKEYCOL, TARGET
REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;
```

Fix ? Adding supplemental logging?

Source

```
add trandata repuser2.testkeycol, nokey,
cols(col1,col2,col3)

select a.* from dba_log_group_columns a
where table_name='TESTKEYCOL';

OWNER        LOG_GROUP_NAME
        TABLE_NAME        COLUMN_NAME
        POSITION     LOGGING_PROPERTY

REPUSER2     GGS_109979        TESTKEYCOL
        COL1 1        LOG

REPUSER2     GGS_109979        TESTKEYCOL
        COL2 2        LOG

REPUSER2     GGS_109979        TESTKEYCOL
        COL3 3        LOG
```

Keycol tables

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

Fix ? Adding supplemental logging?

Source

select rowid,a.* from repuser2.testkeycol a;

ROWID	COL1	COL2	COL3
AAAa2bAAEAAAIRVAAA	1	100	Row1
AAAa2bAAEAAAIRVAAB	2	200	Row2
AAAa2bAAEAAAIRVAAC			Row3
AAAa2bAAEAAAIRVAAD	6	600	Row4
AAAa2bAAEAAAIRVAAE	5	500	Row5
AAAa2bAAEAAAIRXAAB			Row6

delete from repuser2.testkeycol where rowid='AAAa2bAAEAAAIRXAAB';

commit;

GG replication

Target

select rowid,a.* from repuser2.testkeycol a;

ROWID	COL1	COL2	COL3
AAAPFfAAEAAAAK7AAA	1	100	Row1
AAAPFfAAEAAAAK7AAB	2	200	Row2
AAAPFfAAEAAAAK7AAC	6	600	Row4
AAAPFfAAEAAAAK7AAE	5	500	Row5
AAAPFfAAEAAAAK+AAB			Row6

→ row3 was deleted

DELETE FROM "REPUSER2"."TESTKEYCOL" WHERE "COL1" is NULL AND "COL2" is NULL AND ROWNUM = 1

Keycol tables

Table with set of columns (Keycol table)

- Set of columns that will uniquely identify the each row
- Table is small
- Currently no duplicate records exist
- Analyze/Examine batch operations such as betch delete
- **At least one column in keycol columns should be NOT NULL (Same case for unique index)**
- **Make at least one keycol column NOT NULL if all are NULLABLE or have a regular monitoring to alert if there is NULL on keycol columns**

Keycol tables

Source

TABLE repuser2.testkeycol;

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL;

Table with UI

Source

insert into repuser2.testkeycol values(1,100,'Row1');

insert into repuser2.testkeycol values(2,200,'Row2');

insert into repuser2.testkeycol values(null,null,'Row3');

insert into repuser2.testkeycol values(null,null,'Row4');

insert into repuser2.testkeycol values(5,500,'Row5');

Create index repuser2.idxui on repuser2.testkeycol (col1,col2);

GG replication

Target

select * from repuser2.testkeycol

COL1	COL2	COL3
1	100	Row1
2	200	Row2
		Row3
		Row4
5	500	Row5

Keycol tables

Source

TABLE repuser2.testkeycol;

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL;

Table with UI

Source

select rowid,a.* from repuser2.testkeycol a

ROWID	COL1	COL2	COL3
AAAa2bAAEAAAIRVAAA	1	100	Row1
AAAa2bAAEAAAIRVAAB	2	200	Row2
AAAa2bAAEAAAIRVAAC			Row3
AAAa2bAAEAAAIRVAAD			Row4
AAAa2bAAEAAAIRVAAE	5	500	Row5

Updating 4th row

update repuser2.testkeycol set col1=4,col2=400 where rowid='AAAa2bAAEAAAIRVAAD';

Commit;

GG replication

Target

select * from repuser2.testkeycol

COL1	COL2	COL3
1	100	Row1
2	200	Row2
4	400	Row3
		Row4
5	500	Row5

Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- DDL replication and columns
- Replication between different structures
- What specific to monitor
- Additional quick tips
- Upcoming topics

Supplemental logging

Supplemental logging

- Minimum level is required at the database level

  ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;

  logs the minimal amount of information needed for LogMiner to identify, group, and merge the redo operations associated with DML changes"

  GG does not startup (tested in 11.2) if minimum is not set

- Other : Identification key logging (PK, UK, FK)
  - Database level
  - Schema level (GG >=11.1, bug 13794550)
    - Database : e.g. DBMS_CAPTURE_ADM.PREPARE_SCHEMA_INSTANTIATION
    - GoldenGate: e.g. ADD SCHEMATRANDATA <Schema NAME>
  - Table level
    - Method
      - Database : e.g. ALTER TABLE <> ADD SUPPLEMENTAL LOG GROUP Group_name (Column_name) always;
      - GoldenGate : e.g. ADD TRANDATA <Table Name>

Supplemental logging

Source

TABLE repuser2.*;

Target

MAP repuser2.*, TARGET repuser2.*;

No supplemental logging on source

Source

desc repuser2.testsp

ID	NUMBER
FNAME	VARCHAR2(100)
LNAME	VARCHAR2(100)
ISSUED	DATE

alter table repuser2.testsp add primary key(id);

select * from dba_log_group_columns where table_name='TESTSP';

no row → **No supplemental logging**

GG replication

Target

desc repuser2.testsp

ID	NUMBER
FNAME	VARCHAR2(100)
LNAME	VARCHAR2(100)
ISSUED	DATE

select * from dba_log_group_columns where table_name='TESTSP';

no row → **No supplemental logging**

Supplemental logging

Source

TABLE repuser2.*;

Target

MAP repuser2.*, TARGET repuser2.*;

No supplemental logging on source

Source

insert into repuser2.testsp values (1,'Test','TestL1',sysdate);

insert into repuser2.testsp values (2,'Test','TestL2',sysdate);

insert into repuser2.testsp values (3,'Test','TestL3',sysdate);

commit;

GG replication

Target

select rowid,a.* from repuser2.testsp a;

ROWID	ID	FNAME	LNAME	ISSUED
AAAPHzAAEAAAALIAAA	1	Test	TestL1	09/14/2013 08:34:51
AAAPHzAAEAAAALIAAB	2	Test	TestL2	09/14/2013 08:34:51
AAAPHzAAEAAAALIAAC	3	Test	TestL3	09/14/2013 08:34:52

Supplemental logging

Source

TABLE repuser2.*;

Target

MAP repuser2.*, TARGET repuser2.*;

No supplemental logging on source

Source

update repuser2.testsp set lname='TestL3Update' where fname='Test' and id=3;

commit;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3Update**	**09/14/2013 08:34:52**

GG replication

Target

OCI Error ORA-01403: no data found, SQL <UPDATE "REPUSER2"."TESTSP" SET "LNAME" = :a1 WHERE "ID" = :b0>

Operation failed at seqno 49 rba 14584

Discarding record on action DISCARD on error 1403
Problem replicating REPUSER2.TESTSP to REPUSER2.TESTSP

Record not found,Mapping problem with compressed update record (target format)...

*

ID =

LNAME = TestL3Update

Supplemental logging

Source

TABLE repuser2.*;

Target

MAP repuser2.*, TARGET repuser2.*;

Adding supplemental logging on source

Source

ALTER TABLE repuser2.testsp ADD SUPPLEMENTAL LOG group testsplog (id) always;

select owner,table_name,column_name from dba_log_group_columns where table_name='TESTSP';

OWNER	TABLE_NAME	COLUMN_NAME
REPUSER2	**TESTSP**	**ID**

update repuser2.testsp set lname='TestL3Update' where fname='Test' and id=3;

commit;

GG replication

Target

select * from repuser2.testsp a;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3Update**	**09/14/2013 08:34:52**

Supplemental logging

Source

Extract : TABLE repuser2.*;

Replicat : MAP repuser2.*, TARGET repuser2.*;

Target

Replicat : MAP repuser2.*, TARGET repuser2.*;

Extract : TABLE repuser2.*;

supplemental logging on target

Source

OCI Error ORA-01403: no data found, SQL <UPDATE "REPUSER2"."TESTSP" SET "LNAME" = :a1 WHERE "ID" = :b0>

Operation failed at seqno 18 rba 1819,Discarding record on action DISCARD on error 1403

Problem replicating REPUSER2.TESTSP to REPUSER2.TESTSP

Record not found

Mapping problem with compressed update record (target format)...

*

ID =

LNAME = TestL3UpdateFromTarget

GG replication

Target

update repuser2.testsp set lname='TestL3UpdateFromTarget' where fname='Test' and id=3;

commit;

select * from repuser2.testsp a;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3UpdateFromTarget**	**09/14/2013 08:34:52**

Supplemental logging

Source

Extract : TABLE repuser2.*;

Replicat : MAP repuser2.*, TARGET repuser2.*;

Target

Replicat : MAP repuser2.*, TARGET repuser2.*;

Extract : TABLE repuser2.*;

supplemental logging on target

Source

select * from repuser2.testsp a;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3UpdateFromTarget2**	**09/14/2013 08:34:52**

GG replication

Target

ALTER TABLE repuser2.testsp ADD SUPPLEMENTAL LOG group testsplog (id) always;

update repuser2.testsp set lname='TestL3UpdateFromTarget2' where fname='Test' and id=3;

commit;

select * from repuser2.testsp a;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3UpdateFromTarget2**	**09/14/2013 08:34:52**

Supplemental logging

Source

Extract : TABLE repuser2.*;

Replicat : MAP repuser2.*, TARGET repuser2.*;

Target

Replicat : MAP repuser2.*, TARGET repuser2.*;

Extract : TABLE repuser2.*;

supplemental logging on target

Source

select * from repuser2.testsp a;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3UpdateFromTarget3**	**09/14/2013 08:34:52**

GG replication

Target

delete trandata repuser2.testsp;

add schematrandata repuser2;

update repuser2.testsp set lname='TestL3UpdateFromTarget3' where fname='Test' and id=3;

commit;

select * from repuser2.testsp a;

ID	FNAME	LNAME	ISSUED
1	Test	TestL1	09/14/2013 08:34:51
2	Test	TestL2	09/14/2013 08:34:51
3	**Test**	**TestL3UpdateFromTarget3**	**09/14/2013 08:34:52**

Supplemental logging/keycols

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

supplemental logging for keycols

Source

add trandata repuser2.testkeycol, nokey, cols (col1,col2);

select a.* from dba_log_group_columns a where table_name='TESTKEYCOL';

```
OWNER        LOG_GROUP_NAMETABLE_NAME
        COLUMN_NAME      POSITION
        LOGGING_PROPERTY

REPUSER2     GGS_109979 TESTKEYCOL   COL1  1
        LOG

REPUSER2     GGS_109979 TESTKEYCOL   COL2  2
        LOG
```

Target

add trandata repuser2.testkeycol, nokey, cols (col1,col2);

select a.* from dba_log_group_columns a where table_name='TESTKEYCOL';

```
OWNER        LOG_GROUP_NAMETABLE_NAME
        COLUMN_NAME      POSITION
        LOGGING_PROPERTY

REPUSER2     GGS_109979 TESTKEYCOL   COL1  1
        LOG

REPUSER2     GGS_109979 TESTKEYCOL   COL2  2
        LOG
```

Supplemental logging/keycols

Source

TABLE repuser2.testkeycol, keycol(col1,col2);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2) ;

supplemental logging for keycols

Source

insert into repuser2.testkeycol values(1,100,'Row1');

insert into repuser2.testkeycol values(2,200,'Row2');

Select rowid,a.* from repuser2.testkeycol a;

ROWID	COL1	COL2	COL3
AAAa2qAAEAAAIRXAAA	1	100	Row1
AAAa2qAAEAAAIRXAAB	2	200	Row2

update repuser2.testkeycol set col1=22,col2=222,col3='Row22' where rowid='AAAa2qAAEAAAIRXAAB';→ **Updating 2nd row**

Commit;

GG replication

Target

Select * from repuser.keycol;

COL1	COL2	COL3
1	100	Row1
22	222	Row22

UPDATE "REPUSER2"."TESTKEYCOL" SET "COL1" = :a2,"COL2" = :a3,"COL3" = :a4 WHERE "COL1" = :b0 AND "COL2" = :b1 AND ROWNUM = 1;

Supplemental logging/keycols

Source

TABLE repuser2.testkeycol, keycol(col1,col2,col3);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2,col3) ;

supplemental logging on does not match with keycols

Source

add trandata repuser2.testkeycol, nokey, cols (col1,col2);

select a.* from dba_log_group_columns a where table_name='TESTKEYCOL';

```
OWNER        LOG_GROUP_NAMETABLE_NAME
        COLUMN_NAME      POSITION
        LOGGING_PROPERTY

REPUSER2     GGS_109979 TESTKEYCOL   COL1  1
        LOG

REPUSER2     GGS_109979 TESTKEYCOL   COL2  2
        LOG
```

Target

add trandata repuser2.testkeycol, nokey, cols (col1,col2);

select a.* from dba_log_group_columns a where table_name='TESTKEYCOL';

```
OWNER        LOG_GROUP_NAMETABLE_NAME
        COLUMN_NAME      POSITION
        LOGGING_PROPERTY

REPUSER2     GGS_109979 TESTKEYCOL   COL1  1
        LOG

REPUSER2     GGS_109979 TESTKEYCOL   COL2  2
        LOG
```

Supplemental logging/keycols

Source

TABLE repuser2.testkeycol, keycol(col1,col2,col3);

Target

MAP REPUSER2.TESTKEYCOL, TARGET REPUSER2.TESTKEYCOL, KEYCOLS(col1,col2,col3) ;

supplemental logging on does not match with keycols

Source

Select rowid,a.* from repuser2.testkeycol a;

ROWID	COL1	COL2	COL3
AAAa2qAAEAAAIRXAAA	1	100	Row1
AAAa2qAAEAAAIRXAAB	2	200	Row2

update repuser2.testkeycol set col2='Row223' where rowid='AAAa2qAAEAAAIRXAAB'; → **Updating 2nd row**

commit;

GG replication

Target

OCI Error ORA-01403: no data found, SQL <UPDATE "REPUSER2"."TESTKEYCOL" SET "COL1" = :a3,"COL2" = :a4,"COL3" = :a5 WHERE "COL1" = :b0 AND "COL2" = :b1 AND "COL3" is NULL AND ROWNUM = 1>

Mapping problem with compressed key update record (target format)...

COL1 = 2

COL2 = 200

COL3 = NULL

COL1 = 2

COL2 = Row223

COL3 = NULL

Keycol tables

Table with set of columns (Keycol table)

- Set of columns that will uniquely identify the each row
- Table is small
- Currently no duplicate records exist
- Analyze/Examine batch operations such as betch delete
- At least one column in keycol columns should be NOT NULL (Same case for unique index)
- Make at least one keycol column NOT NULL if all are NULLABLE or have a regular monitoring to alert if there is NULL on keycol columns
- **Supplemental logging should be added on both side**
- **It is critical that column list in source and target are matched and supplemental logging on the all keycol columns to be added.**
  - **For each table : Keycol columns in source = keycol columns in target = columns with supplemental logging**

Supplemental logging

Table with Primary key or unique key

- **Supplemental logging**
  - Database level
    - ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
    - ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
  - Schema level
    - exec DBMS_CAPTURE_ADM.PREPARE_SCHEMA_INSTANTIATION('REPUSER2', 'ALLKEYS_ON');
    - Add schematrandata repuser2;
  - Table level
    - ALTER TABLE repuser2.testsp ADD SUPPLEMENTAL LOG group testsplog (id) always;
    - Add trandata repuser2.testsp
    - **Both sides if it is bi-direction replication**

Table with keycols

- **Supplemental logging**
  - Database level
    - ~~ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;~~ (??)
  - Schema level
    - ~~exec DBMS_CAPTURE_ADM.PREPARE_SCHEMA_INSTANTIATION('REPUSER2','ALL');~~ (??)
    - ~~Add schematrandata repuser2~~ **(only works for PK or UK)**
  - Table level
    - ALTER TABLE repuser2.testsp ADD SUPPLEMENTAL LOG group testsplog (column1,column2) always;
    - Add trandata repuser2.testsp, nokey, cols (column1,column2)
    - **Both sides if it is bi-direction replication**

Supplemental logging/DDL

supplemental logging in DDL replication

Source

TABLE repuser2.*

DDL &

INCLUDE ALL OBJNAME REPUSER2.* &

EXCLUDE OBJTYPE SEQUENCE

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Source

create table repuser2.testddl (col1 number not null, col2 varchar2(100) not null, col3 varchar2(100) not null, col4 date);

alter table repuser2.testddl add primary key (col1);

select * from dba_log_group_columns where table_name='TESTDDL';

no rows

GG replication

Target

desc repuser2.testddl

COL1	NOT NULL	NUMBER
COL2	NOT NULL	VARCHAR2(100)
COL3	NOT NULL	VARCHAR2(100)
COL4		DATE

select * from dba_log_group_columns where table_name='TESTDDL';

no rows

Supplemental logging/DDL

supplemental logging in DDL replication

Source

TABLE repuser2.*

DDL &

INCLUDE ALL OBJNAME REPUSER2.* &

EXCLUDE OBJTYPE SEQUENCE

DDLOPTIONS &

ADDTRANDATA &

REPORT

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Source

select * from dba_log_group_columns where table_name='TESTDDL';

```
OWNER        LOG_GROUP_NAMETABLE_NAME
        COLUMN_NAME      POSITION
        LOGGING_PROPERTY

REPUSER2     GGS_110129  TESTDDL        COL1  1
        LOG
```

→ **Supplemental logging is created**

GG replication

Target

select * from dba_log_group_columns where table_name='TESTDDL';

no rows

Supplemental logging/DDL

supplemental logging in DDL replication

Source

TABLE repuser2.*

DDL &

INCLUDE ALL OBJNAME REPUSER2.* &

EXCLUDE OBJTYPE SEQUENCE

DDLOPTIONS &

ADDTRANDATA & -- To add SL to source

GETREPLICATES & -- To capture SL DDL and send it to target

REPORT

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Source

select * from dba_log_group_columns where table_name='TESTDDL';

```
OWNER        LOG_GROUP_NAMETABLE_NAME
        COLUMN_NAME      POSITION
        LOGGING_PROPERTY

REPUSER2     GGS_110129  TESTDDL        COL1  1
        LOG
```

→ **Supplemental logging is created**

GG replication

Target

select * from dba_log_group_columns where table_name='TESTDDL';

no rows

Supplemental logging/DDL

supplemental logging in DDL replication

According to MOS 1472420.1

"If your TRANSLOGOPTIONS EXCLUDEUSER specified in the Extract is the same as the EXTRACT USERID, the DDL to add supplemental logging is not captured and sent to the target."

Do not use the same database user for extract and replicat.
Extract DB USER <> Replicat DB USER

Supplemental logging/DDL

supplemental logging in DDL replication

Source

```
TABLE   repuser2.*
DDL &
INCLUDE ALL OBJNAME REPUSER2.* &
EXCLUDE OBJTYPE SEQUENCE
DDLOPTIONS &
ADDTRANDATA &    -- To add SL to source
GETREPLICATES &    -- To capture SL DDL and send it to target
REPORT
USERID ggadmin, PASSWORD #####
TRANLOGOPTIONS EXCLUDEUSER ggsdb
```

Target

```
MAP REPUSER2.*, TARGET REPUSER2.*;
```

Source

select * from dba_log_group_columns where table_name='TESTDDL';

OWNER	LOG_GROUP_NAME	TABLE_NAME	COLUMN_NAME	POSITION	LOGGING_PROPERTY
REPUSER2	GGS_110129	TESTDDL	COL1	1	LOG

→ **Supplemental logging is created**

GG replication

Target

select * from dba_log_group_columns where table_name='TESTDDL';

OWNER	LOG_GROUP_NAME	TABLE_NAME	COLUMN_NAME	POSITION	LOGGING_PROPERTY
REPUSER2	GGS_110140	TESTDDL	COL1	1	LOG

→ **Supplemental logging is created**

Supplemental logging/DDL

supplemental logging in DDL replication

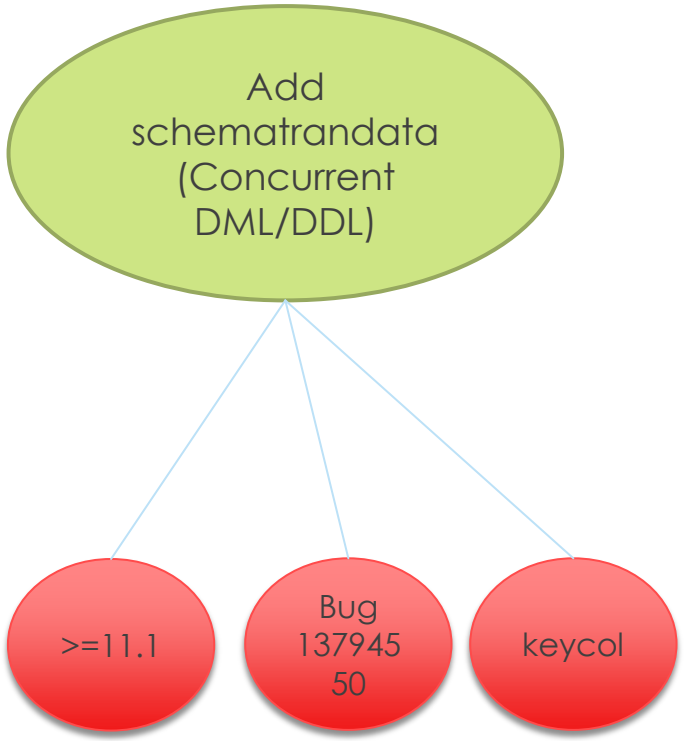
Source

```
TABLE    repuser2.*

DDL &
INCLUDE ALL OBJNAME REPUSER2.* &
EXCLUDE OBJTYPE SEQUENCE

DDLOPTIONS &
ADDTRANDATA &    -- To add SL to source
GETREPLICATES &    -- To capture SL DDL and send it to target
REPORT

USERID extract_user_db, PASSWORD #####
TRANLOGOPTIONS EXCLUDEUSER replicat_user_db
```



Two way replication requires more changes

Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- DDL replication and columns
- Replication between different structures
- What specific to monitor
- Additional quick tips
- Upcoming topics

DDL replication/Columns

Source

TABLE repuser2.*;

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Source

create table repuser2.testcol (id number not null, name varchar2(100) not null);

alter table repuser2.testcol add primary key (id);

GG replication

Target

desc repuser2.testcol

Name	Null?	Type
ID	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2(100)

DDL replication/Columns

Source

TABLE repuser2.*;

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Source

Adding 3 new columns →

alter table repuser2.testcol add (fname varchar2(100) not null);

alter table repuser2.testcol add (salary number);

alter table repuser2.testcol add (debt number);

GG replication

Target

desc repuser2.testcol

Name	Null?	Type
ID	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2(100)
FNAME	NOT NULL	VARCHAR2(100)
DEBT		NUMBER

→ Missing salary column

DDL replication/Columns

Source

TABLE repuser2.*;

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Target

013-09-15 08:34:23 INFO OGG-00482 Oracle GoldenGate Delivery for Oracle, renb_na.prm:DDL found, operation [alter table repuser2.testcol add (fname varchar2(100) not null)(size 65)].

2013-09-15 08:34:23 INFO OGG-00489 Oracle GoldenGate Delivery for Oracle, renb_na.prm: DDL is of mapped scope, after mapping new operation [alter table repuser2."TESTCOL" add (fname varchar2(100) not null)

2013-09-15 08:34:23 INFO OGG-00484 Oracle GoldenGate Delivery for Oracle, renb_na.prm: Executing DDL operation.

2013-09-15 08:34:23 INFO OGG-00483 Oracle GoldenGate Delivery for Oracle, renb_na.prm: DDL operation successful.

2013-09-15 08:35:06 INFO OGG-00484 Oracle GoldenGate Delivery for Oracle, renb_na.prm: Executing DDL operation.

2013-09-15 08:35:06 INFO OGG-00494 Oracle GoldenGate Delivery for Oracle, renb_na.prm: DDL error discarded: error code [DEFAULT], filter [include all (default)], error text [Current time: 2013-09-15 08:35:06

Error text [Error code [54], ORA-00054: resource busy and acquire with NOWAIT specified or timeout expired SQL alter table repuser2."TESTCOL" add (salary number)

/* GOLDENGATE_DDL_REPLICATION */], operation [alter table repuser2."TESTCOL" add (salary number) (size 50)]Operation failed at seqno 53 rba 25196].

2013-09-15 08:35:06 INFO OGG-01408 Oracle GoldenGate Delivery for Oracle, renb_na.prm: Restoring current schema for DDL operation to [ggadmin].

2013-09-15 08:35:57 INFO OGG-00482 Oracle GoldenGate Delivery for Oracle, renb_na.prm: DDL found, operation [alter table repuser2.testcol add (debt number) (size 48)].

2013-09-15 08:35:57 INFO OGG-00484 Oracle GoldenGate Delivery for Oracle, renb_na.prm: Executing DDL operation.

2013-09-15 08:35:57 INFO OGG-00483 Oracle GoldenGate Delivery for Oracle, renb_na.prm: DDL operation successful.

DDL replication/Columns

Source

TABLE repuser2.*;

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Manual fix

Target

Manual run →

alter table repuser2.testcol add (salary number);

Name	Null?	Type
ID	NOT NULL	NUMBER
NAME	NOT NULL	VARCHAR2(100)
FNAME	NOT NULL	VARCHAR2(100)
DEBT		NUMBER
SALARY		NUMBER

DDL replication/Columns

Source

TABLE repuser2.*;

Target

MAP REPUSER2.*, TARGET REPUSER2.*;

Replication

Source

insert into repuser2.testcol
values(1000,'Goerge','Anderson',80000,1000);

commit;

select * from repuser2.testcol;

ID	NAME	FNAME	SALARY	DEBT
1000	Goerge	Anderson	80000	1000

GG replication

Target

select * from repuser2.testcol;

ID	NAME	FNAME	DEBT	SALARY
1000	Goerge	Anderson	80000	1000

DDL replication/Columns

Table with different col. order

Source

```
After  Image:                                  Partition 4   G  s
 0000 0008 0000 0004 3130 3030 0001 000a 0000 0006 |  ........1000........
 476f 6572 6765 0002 000c 0000 0008 416e 6465 7273 |  Goerge........Anders
 6f6e 0003 0009 0000 0005 3830 3030 3000 0400 0800 |  on........80000.....
 0000 0431 3030 30                                 |  ...1000
Column     0 (x0000), Len     8 (x0008)
 0000 0004 3130 3030                               |  ....1000
Column     1 (x0001), Len    10 (x000a)
 0000 0006 476f 6572 6765                          |  ....Goerge
Column     2 (x0002), Len    12 (x000c)
 0000 0008 416e 6465 7273 6f6e                     |  ....Anderson
Column     3 (x0003), Len     9 (x0009)
 0000 0005 3830 3030 30                            |  ....80000
Column     4 (x0004), Len     8 (x0008)
 0000 0004 3130 3030                               |  ....1000
```

1. **Column order between source and target should match**
2. **Replicat should be setup to crash if DDL statement fails**
3. **DDL lock timeout should be set**
4. **Column order between source and target should be compared and any issue should be reported.**
5. **Silent issue or with "Bad Column index" error depends on the situation.**

Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- Replication between different structures
- What specific to monitor
- Additional quick tips
- Upcoming topics

Different structure

Different structure

DB1 (Active – Application release)

Extract :

TABLE repuser2.*

DDL INCLUDE ALL OBJNAME REPUSER2.*

DDLOPTIONS ADDTRANDATA GETREPLICATES REPORT

Replicat :

MAP REPUSER2.*, TARGET REPUSER2.*;

DB2 (Active – User changes)

Extract :

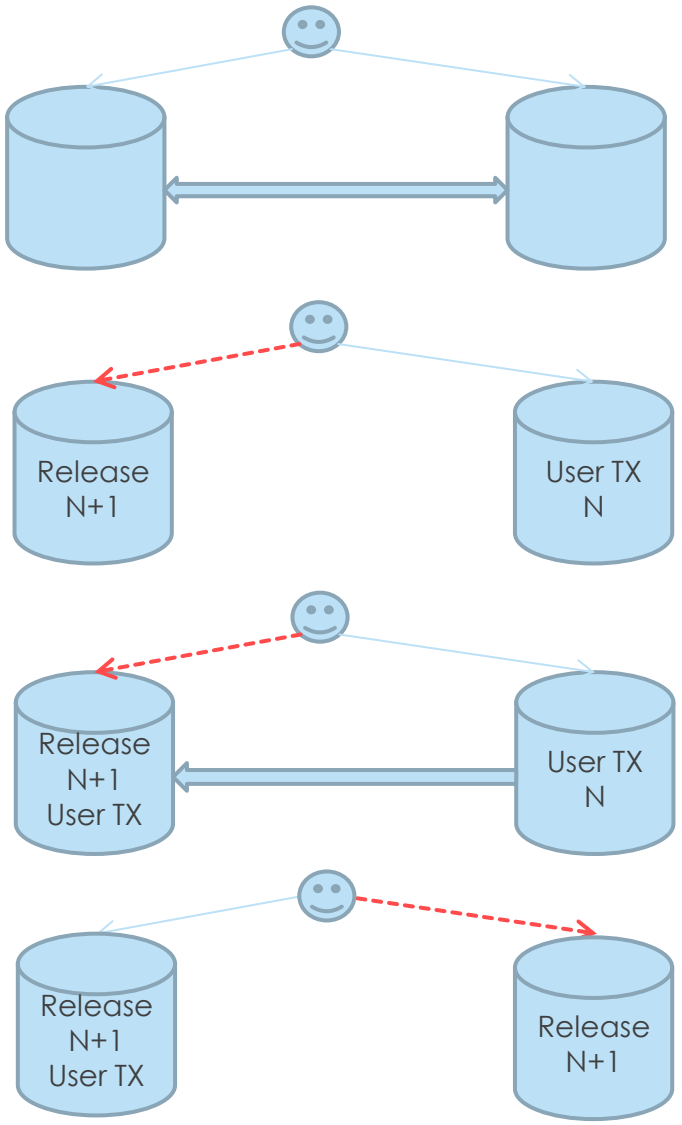
TABLE repuser2.*

Replicat :

MAP REPUSER2.*, TARGET REPUSER2.*;

ASSUMETARGETDEFS (Default)

- **Identical column name**
- **Identical column order**
- **Identical data type**



Different structure

Different structure

Defgen

Utility which comes with Oracle GoldenGate software (No Extra cost !!!)

This tool generates a definition of source tables in a file

Sample Defgen

*

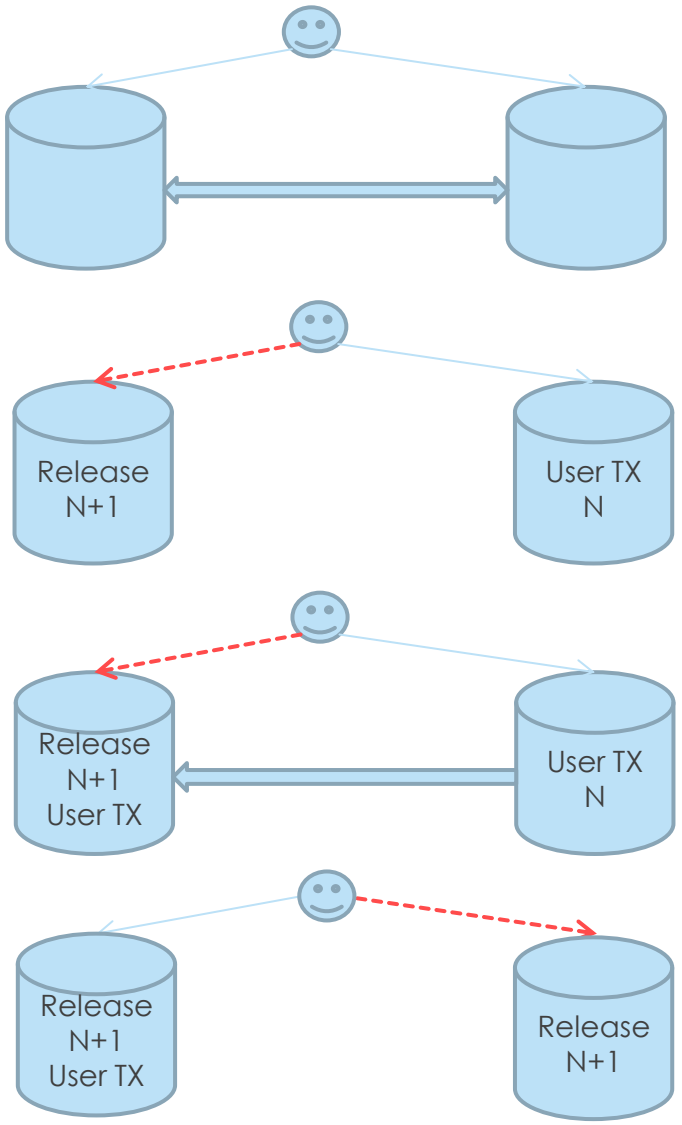
Definition for table REPUSER2.TESTKEYCOL

Record length: 128

Syskey: 0

Columns: 3

COL1	64	50	0 0 0 1 0	50	50	50 0 0 0 0 1	0 0 2
COL2	64	30	56 0 0 1 0	30	30	0 0 0 0 0 1	0 1 0
COL3	64	30	92 0 0 1 0	30	30	0 0 0 0 0 1	0 0 0



Different structure

Table with Structure change

Drop Column

DB1

desc mytab

ID NOT NULL number

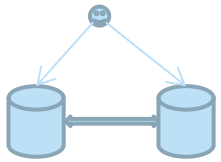
NAME varchar2(10)

DB2

desc mytab

ID NOT NULL number

NAME varchar2(10)



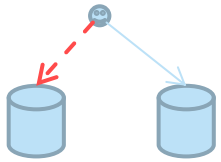
DB1

Alter table mytab drop column name;

DB2

Insert into mytab values(1,'TEST01');

Insert into mytab values(2,'TEST02');



DB1

Replicat :

-- assumetargetdefs

Sourcedefs dirsql/db2defgen.sql

MAP mytab, TARGET mytab,

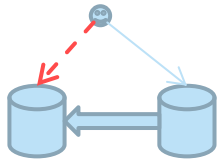
COLMAP (

ID=ID);

DB2

Generate defgen and copy to DB1

Defgen paramfiledefgen.prm → output to db2defgen.sql



DB1

Select * from mytab;

1

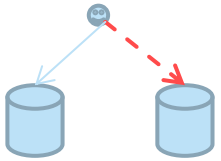
2

DB2

Select * from mytab;

1 TEST01

2 TEST02



Different structure

Table with Structure change

Drop Table

DB1

desc mytab

ID NOT NULL number

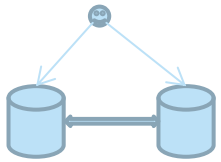
NAME varchar2(10)

DB2

desc mytab

ID NOT NULL number

NAME varchar2(10)



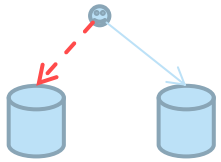
DB1

Drop table mytab;

DB2

Insert into mytab values(1,'TEST01');

Insert into mytab values(2,'TEST02');



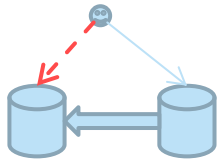
DB1

Replicat :

-- assumetargetdefs

Tabexclude mytab

DB2



DB1

Select * from mytab;

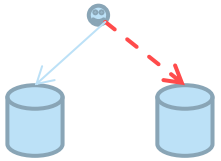
ORA-00942 table or view does not exist

DB2

Select * from mytab;

1 TEST01

2 TEST02



Different structure

Table with Structure change

Modify column

DB1

desc mytab

ID NOT NULL number

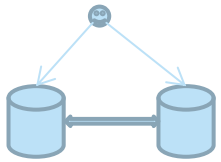
NAME varchar2(10)

DB2

desc mytab

ID NOT NULL number

NAME varchar2(10)



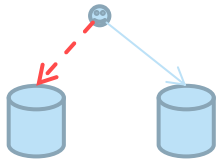
DB1

Alter table mytab modify column name varchar2(6);

DB2

Insert into mytab values(1,'TEST011');

Insert into mytab values(2,'TEST022');



DB1

Replicat :

Sourcedefs dirsql/db2defgen.sql

MAP mytab, TARGET mytab,

COLMAP (

USERDEFAULTS,

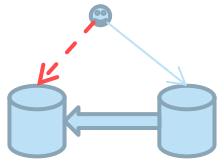
NAME=@STREXT(NAME,1,6)

);

DB2

Generate defgen and copy to DB1

Defgen paramfiledefgen.prm → output to db2defgen.sql



DB1

Select * from mytab;

1 TEST01

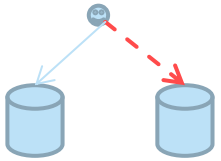
2 TEST02

DB2

Select * from mytab;

1 TEST011

2 TEST022



Different structure

Table with Structure change

Rename column

DB1

desc mytab

ID NOT NULL number

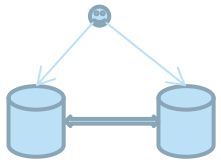
NAME varchar2(10)

DB2

desc mytab

ID NOT NULL number

NAME varchar2(10)



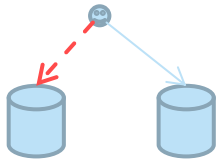
DB1

Alter table mytab rename column name to firstname;

DB2

Insert into mytab values(1,'TEST01');

Insert into mytab values(2,'TEST02');



DB1

Replicat :

Sourcedefs dirsql/db2defgen.sql

MAP mytab, TARGET mytab,

COLMAP (

USERDEFAULTS,

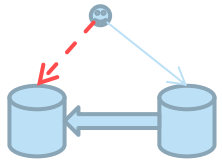
FIRSTNAME=NAME

);

DB2

Generate defgen and copy to DB1

Defgen paramfiledefgen.prm → output to db2defgen.sql



DB1

Select * from mytab;

1 TEST01

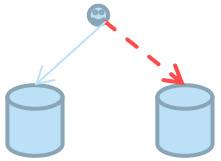
2 TEST02

DB2

Select * from mytab;

1 TEST01

2 TEST02



Different structure

Table with Structure change

Rename table

DB1

desc mytab

ID NOT NULL number

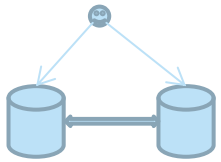
NAME varchar2(10)

DB2

desc mytab

ID NOT NULL number

NAME varchar2(10)



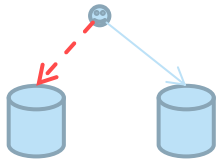
DB1

Alter table mytab rename mynewtab;

DB2

Insert into mytab values(1,'TEST01');

Insert into mytab values(2,'TEST02');

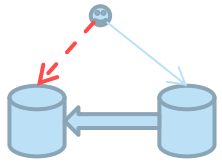


DB1

Replicat :

MAP mytab, target mynewtab;

DB2



DB1

Select * from mynewtab;

1 TEST01

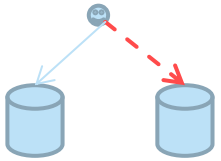
2 TEST02

DB2

Select * from mytab;

1 TEST01

2 TEST02



Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- Replication between different structures
- What specific to monitor
- Additional quick tips
- Upcoming topics

What specific to monitor?

Basic health/function

- GG processes are running
- GG lag of processes
- Resource usage (CPU/IO)

Discard file

- Alert any new discard entry
- Keep history of any discard issue
- Pipe to a table and run real time query for new and historical

Different data

- Table comparison between source and target.
- Oracle Veridata tool or script
- Built in incremental feature if possible

Null on keycol columns

- If table
  - **Uses keycols &**
  - **Receives Update/delete**

Nullable keycol columns should be monitored for any NULL value

Column order

- Order of columns in source and target database should be identical. (as long as there is not mapping/transformation)

Supplemental logging

- Supplemental logging on DB is must.
- Supplemental logging for keycols columns should exist and should be identical between source and target.

Topics

- First step – Where to start
- Tables
  - Tips on parameter files and replicated table
  - Tables with no PK/UI
  - Nullable colums
- Supplemental logging
  - When it is required
  - Supplemental logging and keycols
  - Supplemental logging and DDL
- Replication between different structures
- What specific to monitor
- Additional small quick tips
- Upcoming topics

Just some quick tips

DDL vs Truncate

- No DDL does not mean no truncate in various places
- Be specific if truncate is used
- GETTRUNCATES

Parameter

- Make sure always set NLS_LANG in parameter files
- Do not ignore any warning in ggserr.log

Classic vs Integrated

- Utilize classic GoldenGate unless there is a show stopper

Replicat error

- If replicat encounters an error, it should be stopped and should not skip transaction.
- No discard for replicat

Stats

- Before any major change, reset stats
- By default stats are reported since startup a process

Golden DBA-Upcoming topics

Golden DBA
-Advanced configuration setup

Golden DBA
- Performance tuning- Solutions/Issues

Golden DBA
- Deep dive troubleshooting

Golden DBA
-Two way replication and conflict resolution

Questions