Oracle GoldenGate - DDL Replication Objects

The following page describes the objects created for Oracle GoldenGate DDL support.

The page is based on Oracle GoldenGate version 11.2.1.0.1. For a basic configuration I used two Linux VMs (OEL5U6) running single instance Oracle 11.2.0.3 databases. I created both databases using DBCA.

The motivation behind this page was to determine the source from which GoldenGate extracted DDL. Options were the online redo log or DDL triggers. The conclusion of these investigations is that GoldenGate uses a DDL trigger to capture DDL changes. It supplements this information with data collected from various tables and views including NLS_SESSION_PARAMETERS.

Note that it is possible for aliases to be created for all DDL replication tables. However, there does not appear to be any real benefit in doing in a production environment. Therefore this page refers to all tables by their default names.

The main tables used during DDL replication are:

- GGS_MARKER
- GGS_DDL_SEQ

Tables

GGS_MARKER

The GGS_MARKER table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
FRAGMENTNO	NUMBER
OPTIME	CHAR(19)
TYPE	VARCHAR2(100)
SUBTYPE	VARCHAR2(100)
MARKER_TEXT	VARCHAR2(4000)

This table has the following indexes:

Index Name	Column Name
GGS_MARKER_IND1	SEQNO
	FRAGMENTNO
SYS_Cnnnnnn	OPTIME
	SEQNO
	FRAGMENTNO

This table captures all DDL executed in the source database including DDL for Oracle objects (such as the AWR).

Each DDL statement is allocated a new sequence number from the GGS_MARKER_SEQ sequence.

Each DDL statement appears to occupy two rows in the GGS_MARKER table with the following fragment numbers:

- Fragment 1 contains the DDL statement in the MARKER_TEXT column.
- Fragment 2 contains the marker environment.

For example, consider the following statement:

```
ALTER TABLE t100 ADD c4 NUMBER
```

In this example, the sequence number allocated was 501.

The DDL statement is stored in the MARKER_TEXT column of the first fragment:

```
,C1='ALTER TABLE t100 ADD c4 NUMBER ',
```

The commas appear to be used as delimiters.

The environment is stored in the MARKER_TEXT column of the second fragment. The following example has been reformatted and commented for readability:

```
,C5='501',# GGS_MARKER Sequence Number
,B2='77002',# Object ID
,G4='',# Sequence ROWID?
,B3='US01',             # Object Owner
,B4='T100',# Object Name
,C12='',# Master Owner
,C13='',# Master Object
,B5='TABLE',# Object Type
,B6='ALTER',# DDL Type
,B7='501',# DDL_HIST Sequence Number
,B8='GG01.GGS_DDL_HIST',# DDL_HIST table name
,B9='US01',# Login User Name
,C7='11.2.0.3.0',# Table Version
,C8='11.2.0.0.0',# Table Version - Compatible
,C9='VALID',# Table Validity
,C10='1',# Instance Number
,C11='NORTH',# Instance Name
,G3='NONUNIQUE',# Is Index Unique?
,C14='NO',# Object Table?
,C20='NO',# XML Type Table?
,C17('1')='NLS_LANGUAGE',# NLS Parameter Name
,C18('1')='AMERICAN',# NLS Parameter Value
,C17('2')='NLS_TERRITORY',              # NLS Parameter Name
,C18('2')='AMERICA',# NLS Parameter Value
,C17('3')='NLS_CURRENCY',# NLS Parameter Name
,C18('3')='$',# NLS Parameter Value
,C17('4')='NLS_ISO_CURRENCY',# NLS Parameter Name
,C18('4')='AMERICA',# NLS Parameter Value
,C17('5')='NLS_NUMERIC_CHARACTERS',# NLS Parameter Name
,C18('5')='.\\,',# NLS Parameter Value
,C17('6')='NLS_CALENDAR',# NLS Parameter Name
,C18('6')='GREGORIAN',# NLS Parameter Value
,C17('7')='NLS_DATE_FORMAT',# NLS Parameter Name
,C18('7')='DD-MON-RR',# NLS Parameter Value
,C17('8')='NLS_DATE_LANGUAGE',# NLS Parameter Name
,C18('8')='ENGLISH',# NLS Parameter Value
,C17('9')='NLS_SORT',# NLS Parameter Name
,C18('9')='BINARY',# NLS Parameter Value
,C17('10')='NLS_TIME_FORMAT',# NLS Parameter Name
,C18('10')='HH.MI.SSXFF AM',# NLS Parameter Value
,C17('11')='NLS_TIMESTAMP_FORMAT',# NLS Parameter Name
,C18('11')='DD-MON-RR HH.MI.SSXFF AM',# NLS Parameter Value
,C17('12')='NLS_TIME_TZ_FORMAT',# NLS Parameter Name
,C18('12')='HH.MI.SSXFF AM TZR',# NLS Parameter Value
,C17('13')='NLS_TIMESTAMP_TZ_FORMAT',# NLS Parameter Name
,C18('13')='DD-MON-RR HH.MI.SSXFF AM TZR',# NLS Parameter Value
,C17('14')='NLS_DUAL_CURRENCY',# NLS Parameter Name
,C18('14')='$',# NLS Parameter Value
,C17('15')='NLS_COMP',# NLS Parameter Name
,C18('15')='BINARY',# NLS Parameter Value
,C17('16')='NLS_LENGTH_SEMANTICS',# NLS Parameter Name
,C18('16')='BYTE',# NLS Parameter Value
,C17('17')='NLS_NCHAR_CONV_EXCP',# NLS Parameter Name
```

```
,C18('17')='FALSE',# NLS Parameter Value
,C19='17',# Number of NLS parameters
```

The NLS parameter names and values are determined from NLS_SESSION_PARAMETERS using the following query:

```
SELECT parameter, value
FROM nls_session_parameters;
```

We can conclude from the above NLS parameters with a reasonable level of certainty that GoldenGate does not use the values stored in the Redo Operation 24.1 (DDL) records.

GGS_DDL_HIST

The GGS_DDL_HIST table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
OBJECTID	NUMBER
DATAOBJECTID	NUMBER
DDLTYPE	VARCHAR2(40)
OBJECTNAME	VARCHAR2(100)
OBJECTOWNER	VARCHAR2(100)
OBJECTTYPE	VARCHAR2(40)
FRAGMENTNO	NUMBER
OPTIME	CHAR(19)
STARTSCN	NUMBER
METADATA_TEXT	VARCHAR2(4000)
AUDITCOL	VARCHAR2(80)

This table has the following indexes:

Index Name	Column Name
GGS_DDL_HIST_INDEX1	OBJECTID
GGS_DDL_HIST_i1	SEQNO
	FRAGMENTNO
GGS_DDL_HIST_i2	OBJECTID
	STARTSCN
	FRAGMENTNO
GGS_DDL_HIST_i3	STARTSCN
	FRAGMENTNO
GGS_DDL_HIST_i4	OBJECTNAME
	OBJECTOWNER
	OBJECTTYPE
	STARTSCN
	FRAGMENTNO
GGS_DDL_HIST_i5	OPTIME
GGS_DDL_HIST_i6	STARTSCN
	AUDITCOL
	FRAGMENTNO

This table contains all DDL statements executed for non-Oracle objects, irrespective of whether they are being replicated or not.

The table captures ALTER and DROP DDL, but does not capture CREATE DDL.

The GGS_DDL_HIST table is similar to the GGS_MARKER table in many ways.

Each captured DDL statement is allocated a new sequence number from the GGS_MARKER_SEQ sequence.

Each captured DDL statement appears to occupy two rows in the GGS_MARKER table with the following fragment numbers:

- Fragment 1 contains the DDL statement in the METADATA_TEXT column.
- Fragment 2 contains the metadata in the METADATA_TEXT column:

The following example uses table T100 which was created as follows:

```
CREATE TABLE t100
(
  c1 NUMBER,
  c2 VARCHAR2(30),
  c3 DATE
);
```

The above statement is not captured in the GGS_DDL_HIST table.

Consider the following statement:

```
ALTER TABLE t100 ADD c4 NUMBER
```

The above statement is captured in the GGS_DDL_HIST table.

In this example, the sequence number allocated was 501.

The DDL statement is stored in the METADATA_TEXT column of the first fragment:

```
,G1='ALTER TABLE t100 ADD c4 NUMBER ',
```

Again the commas appear to be used as delimiters.

The environment is stored in the METADATA_TEXT column of the second fragment. The following example has been reformatted and commented for readability:

```
,C6='GG01.GGS_MARKER',# Marker Table Name
,C5='501',# Marker Sequence Number
,C2='1885306',# SCN
,S='77002',# Object ID
,W='US01',# Owner
,X='T100',# Name
,Y='TABLE',# Object Type
,Z='ALTER',# Operation Type
,A1='84',# User ID
,A1='84',# User ID
,C3='',# Master Owner
,C4='',# Master Name
,C15='NO',# Ignore?
,R='8192',# Block Size
,I='77002',# Data Object ID
,J='',# Cluster Columns
,K='3',# Total Number of Columns
,L='0',# Log Group Exists
,N='VALID',# Valid?
,G11='FALSE',# Cluster?
,G7='NO',# IOT?
,G8='NO',# IOT Overflow?
,O='NO',# Subpartitioned?
,P='NO',# Partitioned?
```

```
# Column C1 definitions
,C('1'),C('C1')='C1',# Alternative Name
,D('1'),D('C1')='NUMBER',# Alternative Type
,G13('1'),G13('C1')='',# Alternative XML Type
,E('1'),E('C1')='',# Alternative Precision
,F('1'),F('C1')='',# Alternative Char Used
,G10('1'),G10('C1')='22',# Alternative Length
,G20('1'),G20('C1')='NO',# Is Encrypted?
,G22('1'),G22('C1')='NO',# Is LOB?
,G21('1'),G21('C1')='NO',# No Salt?
,G27('1'),G27('C1')='NO',# Has Not Null Default?
,G('1'),G('C1')='0',# Alternative XML Type
,G9('1'),G9('C1')='0',# Alternative BinaryXML Type
,G26('1'),G26('C1')='0',# Alternative ObjectXML Type
,A2('1'),A2('C1')='C1',# Column Name
,A3('1'),A3('C1')='1',# Column Number
,A4('1'),A4('C1')='1',# Segment Column Number
,A5('1'),A5('C1')='2',# Column Type (2=NUMBER)
,A6('1'),A6('C1')='22',# Length
,A7('1'),A7('C1')='1',# Is NULL?
,A8('1'),A8('C1')='',# Precision
,A9('1'),A9('C1')='',# Scale
,B1('1'),B1('C1')='0',# Character Set ID
,A('1'),A('C1')='0',# Character Set Form

# Column C2 definitions
,C('2'),C('C2')='C2',# Alternative Name
,D('2'),D('C2')='VARCHAR2',# Alternative Type
,G13('2'),G13('C2')='',# Alternative XML Type
,E('2'),E('C2')='',# Alternative Precision
,F('2'),F('C2')='B',# Alternative Char Used
,G10('2'),G10('C2')='30',# Alternative Length
,G20('2'),G20('C2')='NO',# Is Encrypted?
,G22('2'),G22('C2')='NO',# Is LOB?
,G21('2'),G21('C2')='NO',# No Salt?
,G27('2'),G27('C2')='NO',# Has Not NULL Default?
,G('2'),G('C2')='0',# Alternative XML Type
,G9('2'),G9('C2')='0',# Alternative BinaryXML Type
,G26('2'),G26('C2')='0',# Alternative ObjectXML Type
,A2('2'),A2('C2')='C2',# Column Name
,A3('2'),A3('C2')='2',# Column Number
,A4('2'),A4('C2')='2',# Segment Column Number
,A5('2'),A5('C2')='1',# Column Type (1=VARCHAR2)
,A6('2'),A6('C2')='30',# Length
,A7('2'),A7('C2')='1',# Is NULL?
,A8('2'),A8('C2')='',# Precision
,A9('2'),A9('C2')='',# Scale
,B1('2'),B1('C2')='178',# Character Set ID
,A('2'),A('C2')='1',# Character Set Form

# Column 3 definitions
,C('3'),C('C3')='C3',# Alternative Name
,D('3'),D('C3')='DATE',# Alternative Type
,G13('3'),G13('C3')='',# Alternative XML Type
,E('3'),E('C3')='',# Alternative Precision
,F('3'),F('C3')='',# Alternative Char Used
,G10('3'),G10('C3')='7',# Alternative Length
,G20('3'),G20('C3')='NO',# Is Encrypted?
,G22('3'),G22('C3')='NO',# Is LOB?
,G21('3'),G21('C3')='NO',# No Salt?
,G27('3'),G27('C3')='NO',# Has Not NULL Default?
,G('3'),G('C3')='0',# Alternative XML Type
,G9('3'),G9('C3')='0',# Alternative BinaryXML Type
,G26('3'),G26('C3')='0',# Alternative ObjectXML Type
,A2('3'),A2('C3')='C3',# Column Name
,A3('3'),A3('C3')='3',# Column Number
,A4('3'),A4('C3')='3',# Segment Column Number
```

```
,A5('3'),A5('C3')='12',# Column Type (12=DATE)
,A6('3'),A6('C3')='7',# Length
,A7('3'),A7('C3')='1',# Nullable
,A8('3'),A8('C3')='',# Precision
,A9('3'),A9('C3')='',# Scale
,B1('3'),B1('C3')='0',# Character Set ID
,A('3'),A('C3')='0',# Character Set Form

,H='3',# Column Count (=3)
,G28='NO',# XML Type Table?
```

GGS_DDL_HIST_ALT

The GGS_DDL_HIST_ALT table contains the following columns:

Column Name	Data Type
ALTOBJECT_ID	NUMBER
OBJECTID	NUMBER
OPTIME	CHAR(19)

This table has the following indexes:

Index Name	Column Name
GGS_DDL_HIST_ALT_u1	OBJECTID
	ALTOBJECTID
GGS_DDL_HIST_ALT_u2	OPTIME
GGS_DDL_HIST_ALT_u3	ALTOBJECTID
	OBJECTID

GGS_DDL_OBJECTS

The GGS_DDL_OBJECTS table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
OPTIME	CHAR(19)
MARKER_TABLE	VARCHAR2(100)
MARKER_SEQ	NUMBER
START_SCN	NUMBER
OPTYPE	VARCHAR2(40)
OBJTYPE	VARCHAR2(40)
DB_BLOCKSIZE	NUMBER
OBJOWNER	VARCHAR2(100)
OBJNAME	VARCHAR2(100)
OBJECTID	NUMBER
MASTER_OWNER	VARCHAR2(100)
MASTER_NAME	VARCHAR2(100)
DATA_OBJECTID	NUMBER
VALID	VARCHAR2(30)
CLUSTER_COLS	NUMBER
LOG_GROUP_EXISTS	VARCHAR2(20)
SUBPARTITION	VARCHAR2(20)
PARTITION	VARCHAR2(20)
PRIMARY_KEY	VARCHAR2(100)
TOTAL_COLS	NUMBER
COLS_COUNT	NUMBER

DDL_STATEMENT	CLOB

This table does not have any regular indexes. However it does have a LOB index for the DDL_STATEMENT column.

This table is used by DUMPDDL for debugging.

GGS_DDL_PARTITIONS

The GGS_DDL_PARTITIONS table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
PARTITION_ID	NUMBER

This table does not have any indexes.

This table is used by DUMPDDL for debugging.

GGS_DDL_COLUMNS

The GGS_DDL_COLUMNS table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
NAME	VARCHAR2(100)
POS	NUMBER
TYPE	VARCHAR2(40)
LENGTH	NUMBER
ISNULL	VARCHAR2(30)
PREC	NUMBER
SCALE	NUMBER
CHARSETID	VARCHAR2(30)
CHARSETFORM	VARCHAR2(50)
SEGPOS	NUMBER
ALTNAME	VARCHAR2(100)
ALTTYPE	VARCHAR2(40)
ALTPREC	NUMBER
ALTCHARUSED	VARCHAR2(50)
ALTXMLTYPE	VARCHAR2(50)

This table does not have any regular indexes.

This table is used by DUMPDDL for debugging.

GGS_DDL_PRIMARY_KEYS

The GGS_DDL_PRIMARY_KEYS table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
COLUMN_NAME	VARCHAR2(100)

This table does not have any indexes.

This table is used by DUMPDDL for debugging.

GGS_DDL_LOG_GROUPS

The GGS_DDL_LOG_GROUPS table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
COLUMN_NAME	VARCHAR2(100)

Note - this table does have the same definition as GGS_DDL_COLUMNS - It is not a cut-and-paste error.

This table does not have any indexes.

This table is used by DUMPDDL for debugging.

GGS_DDL_RULES

The GGS_DDL_RULES table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
OBJ_NAME	VARCHAR2(200)
OWNER_NAME	VARCHAR2(200)
BASE_OBJ_NAME	VARCHAR2(200)
BASE_OWNER_NAME	VARCHAR2(200)
BASE_OBJ_PROPERTY	NUMBER
OBJ_TYPE	NUMBER
COMMAND	VARCHAR2(50)
INCLUSION	NUMBER

This table has the following index:

Index Name	Column Name
SYS_Cnnnnnn	SNO

GGS_DDL_RULES_LOG

The GGS_DDL_RULES_LOG table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
OBJ_NAME	VARCHAR2(200)
OWNER_NAME	VARCHAR2(200)
BASE_OBJ_NAME	VARCHAR2(200)
BASE_OWNER_NAME	VARCHAR2(200)
BASE_OBJ_PROPERTY	NUMBER
OBJ_TYPE	NUMBER
COMMAND	VARCHAR2(50)

This table does not have any indexes.

GGS_SETUP

The GGS_SETUP table contains the following columns:

Column Name	Data Type
PROPERTY	VARCHAR2(100)
VALUE	VARCHAR2(4000)

In GoldenGate 11.2.1.0.1, GGS_SETUP is created with six initial parameters as follows:

Parameter Name	Value
ALLOWNONVALIDATEDKEYS	0
DDL_SQL_TRACING	0
DDL_STAYMETADATA	OFF
DDL_TRACE_LEVEL	0
_LIMIT32K	0
_USEALLKEYS	0

This table has the following index:

Index Name	Column Name
GGS_SETUP_UKEY	PROPERTY

GGS_STICK

The GGS_STICK table contains the following columns:

Column Name	Data Type
PROPERTY	VARCHAR2(100)
VALUE	VARCHAR2(100)

In GoldenGate 11.2.1.0.1, GGS_STICK is initially created as an empty table.

This table has the following index:

Index Name	Column Name
SYS_Cnnnnnn	PROPERTY

GGS_TEMP_COLS

The GGS_TEMP_COLS table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
COLNAME	VARCHAR2(100)
NULLABLE	NUMBER
VIRTUAL	NUMBER
UDT	NUMBER
ISSYS	NUMBER

This table has the following index:

Index Name	Column Name
SYS_Cnnnnnn	SEQNO
	COLNAME

GGS_TEMP_UK

The GGS_TEMP_UK table contains the following columns:

Column Name	Data Type
SEQNO	NUMBER
KEYNAME	VARCHAR2(100)
COLNAME	VARCHAR2(100)
NULLABLE	NUMBER
VIRTUAL	NUMBER
UDT	NUMBER
ISSYS	NUMBER

This table has the following index:

Index Name	Column Name
SYS_Cnnnnnn	SEQNO
	KEYNAME
	COLNAME

Sequences

GGS_MARKER_SEQ

This sequence is used to generate primary keys for the GGS_MARKER table

This sequence is created by marker_setup.sql

GGS_DDL_SEQ

This sequence is used to generate primary keys for the GGS_DDL_HIST table

This sequence is created by ddl_setup.sql

Packages

DDLREPLICATION

This package is owned by GGSCHEMA.

This package is created by ddl_setup.sql.

The package declares metadata and marker constants

Metadata constants declared are:

Name	Value
MD_TAB_USERID	A1
MD_COL_NAME	A2
MD_COL_NUM	A3
MD_COL_SEGCOL	A4
MD_COL_TYPE	A5
MD_COL_LEN	A6
MD_COL_ISNULL	A7
MD_COL_PREC	A8
MD_COL_SCALE	A9
MD_COL_CHARSETID	B1
MD_COL_CHARSETFORM	A
MD_COL_ALT_NAME	C

MD_COL_ALT_TYPE	D
MD_COL_ALT_PREC	E
MD_COL_ALT_CHAR_USED	F
MD_COL_ALT_XML_TYPE	G
MD_TAB_COLCOUNT	H
MD_TAB_DATAOBJECTID	I
MD_TAB_CLUCOLS	J
MD_TAB_TOTAL_COL_NUM	K
MD_TAB_LOG_GROUP_EXISTS	L
MD_COL_ALT_LOG_GROUP_COL	M
MD_TAB_VALID	N
MD_TAB_SUBPARTITION	O
MD_TAB_PARTITION	P
MD_TAB_PARTITION_IDS	Q
MD_TAB_BLOCKSIZE	R
MD_TAB_OBJECTID	S
MD_TAB_PRIMARYKEY	T
MD_TAB_PRIMARYKEYNAME	V
MD_TAB_OWNER	W
MD_TAB_NAME	X
MD_TAB_OBJTYPE	Y
MD_TAB_OPTYPE	Z
MD_TAB_SCN CONSTANT	C2
MD_TAB_MASTEROWNER	C3
MD_TAB_MASTERNAME	C4
MD_TAB_MARKERSEQNO	C5
MD_TAB_MARKERTABLENAME	C6
MD_TAB_DDLSTATEMENT	G1
MD_TAB_BIGFILE	G2
MD_TAB_ISINDEXUNIQUE	G3
MD_TAB_SEQUENCEROWID	G4
MD_TAB_SEQCACHE	G5
MD_TAB_SEQINCREMENTBY	G6
MD_TAB_IOT CONSTANT	G7
MD_TAB_IOT_OVERFLOW	G8
MD_COL_ALT_BINARYXML_TYPE	G9
MD_COL_ALT_LENGTH	G10
MD_TAB_CLUSTER	G11
MD_TAB_CLUSTER_COLNAME	G12
MD_COL_ALT_TYPE_OWNER	G13
MD_TAB_SESSION_OWNER	G14
MD_TAB_ENC_MKEYID	G15
MD_TAB_ENC_ENCALG	G16
MD_TAB_ENC_INTALG	G17
MD_TAB_ENC_COLKLC	G18
MD_TAB_ENC_KLCLEN	G19
MD_COL_ENC_ISENC	G20
MD_COL_ENC_NOSALT	G21
MD_COL_ENC_ISLOB	G22
MD_COL_LOB_ENCRYPT	G23
MD_COL_LOB_COMPRESS	G24
MD_COL_LOB_DEDUP	G25
MD_COL_ALT_OBJECTXML_TYPE	G26
MD_COL_HASNOTNULLDEFAULT	G27
MD_TAB_XMLTYPETABLE	G28

The above constants are used in GGS_DDL_SEQ.METADATA_TEXT.

Marker constants declared are:

Name	Value
MK_OBJECTID	B2
MK_OBJECTOWNER	B3
MK_OBJECTNAME	B4
MK_OBJECTTYPE	B5
MK_DDLTYPE	B6
MK_DDLSEQ	B7
MK_DDLHIST	B8
MK_LOGINUSER	B9
MK_DDLSTATEMENT	C1
MK_TAB_VERSIONINFO	C7
MK_TAB_VERSIONINFOCOMPAT	C8
MK_TAB_VALID	C9
MK_INSTANCENUMBER	C10
MK_INSTANCENAME	C11
MK_MASTEROWNER	C12
MK_MASTERNAME	C13
MK_TAB_OBJECTTABLE	C14
MK_TAB_TOIGNORE	C15
MK_TAB_NLS_PARAM	C17
MK_TAB_NLS_VAL	C18
MK_TAB_NLS_CNT	C19
MK_TAB_XMLTYPETABLE	C20

The above constants are used in GGS_MARKER.MARKER_TEXT.

This package declares the following procedures and functions:

- PROCEDURE setCtxInfo
- PROCEDURE getObjTypeName
- PROCEDURE getObjType
- PROCEDURE getDDLObjInfo
- PROCEDURE getDDLBaseObjInfo
- PROCEDURE getKeyCols
- PROCEDURE getKeyColsUseAllKeys
- PROCEDURE saveSeqInfo
- PROCEDURE getColDefs
- PROCEDURE getTableInfo
- PROCEDURE insertToMarker
- FUNCTION itemHeader
- FUNCTION getDDLText
- FUNCTION isRecycle
- PROCEDURE getVersion
- PROCEDURE beginHistory
- PROCEDURE endHistory
- PROCEDURE setTracing
- FUNCTION replace_string
- FUNCTION escape_string
- PROCEDURE saveMarkerDDL
- FUNCTION trace_header_name
- FUNCTION removeSQLcomments
- PROCEDURE getTableFromIndex

- PROCEDURE getObjectTableType
- PROCEDURE DDLtooLarge

DDLAUX

This package is owned by GGSCHEMA. It handles INCLUDE and EXCLUDE rules.

The package is created by ddl_setup.sql

This package declares the following procedures and functions:

- FUNCTION addRule
- FUNCTION dropRule
- PROCEDURE listRules
- FUNCTION SKIP_OBJECT
- PROCEDURE recordExclusion

DDLVERSIONSPECIFIC

This package is owned by GGSCHEMA

This package declares a couple of cursors based on data dictionary views. It does not have a body

As the name suggests, the package is version specific. It is created by one of the following scripts:

- ddl_ora9.sql
- ddl_ora10.sql
- ddl_ora11.sql

DDLCTXINFO

This package is owned by SYS. It is created by ddl_setup.sql

Procedures

INITIAL_SETUP

This procedure is created by ddl_setup.sql

It creates the objects required for GGSCI, tracing etc.

Tables created include:

- GGS_DDL_RULES
- GGS_DDL_RULES_LOG
- GGS_TEMP_COLS
- GGS_TEMP_UK
- GGS_STICK
- GGS_SETUP
- GGS_DDL_HIST_ALT
- GGS_DDL_HIST
- GGS_DDL_COLUMNS
- GGS_DDL_LOG_GROUPS
- GGS_DDL_PARTITIONS
- GGS_DDL_PRIMARY_KEYS
- GGS_DDL_OBJECTS

Sequences created include:

- GGS_DDL_SEQ

CREATE_TRACE

This procedure is created by ddl_setup.sql and creates the tracing environment for DDL replication.

All tracing code is created outside of the DDL packages in order to allow package creation to be traced

This procedure determines the value of USER_DUMP_DEST from V$PARAMETER and creates the GGS_DDL_TRACE directory in this location.

CLEAR_TRACE

This procedure is created by ddl_setup.sql and deletes the trace file from the GGS_DDL_TRACE directory.

TRACE_PUT_LINE

This procedure is created by ddl_setup.sql and writes a line to the trace file. If the line length exceeds the output line size, it will be split into multiple lines. If the line exceeds 1000 bytes, it will be split into multiple recursive calls to PUT_LINE.

DDLORA_GETLOBS

This procedure is version specific. It is created by one of the following scripts:

- ddl_ora9.sql
- ddl_ora10.sql
- ddl_ora11.sql

Functions

DDLORA_GETERRORSTACK

This function is only available in Oracle 10.1 and above. It is declared in ddl_ora10upCommon.sql for all releases.

It calls DBMS_UTILITY.FORMAT_ERROR_BACKTRACE to generate the error stack. It returns the last 5000 characters of the error stack to the caller.

DDLORA_ERRORISUSERCANCEL

This function is only available in Oracle 10.1 and above. It is declared in ddl_ora10upCommon.sql for all releases.

It calls DBMS_UTILITY.FORMAT_ERROR_STACK and checks for the following error:

```
ORA-01013 user requested cancel of current operation
```

DDLORA_GETALLCOLSLOGGING

This function is only available in Oracle 10.1 and above. It is declared in ddl_ora10upCommon.sql for all releases.

It reports where a table has *all column logging* enabled for supplemental logging.

Internally it joins the data dictionary objects (OBJ$) and constraint definitions (CDEF$) tables for constraint type 17 (All column supplemental logging)

DDLORA_VERIFYDDL

This function is declared in ddl_setup.sql. It verifies that all objects required for GoldenGate DDL support have been created successfully by checking the dictionary views.

Objects checked include:

Object Name	Object Type	Dictionary View
DDLORA_GETERRORSTACK	FUNCTION	DBA_ERRORS
CREATE_TRACE	PROCEDURE	DBA_ERRORS
TRACE_PUT_LINE	PROCEDURE	DBA_ERRORS
FILE_SEPARATOR	FUNCTION	DBA_ERRORS
INITIAL_SETUP	PROCEDURE	DBA_ERRORS
DDLORA_GETLOBS	PROCEDURE	DBA_ERRORS
DDLORA_GETALLCOLSLOGGING	PROCEDURE	DBA_ERRORS
DDLREPLICATION	PACKAGE	DBA_ERRORS
DDLVERSIONSPECIFIC	PACKAGE	DBA_ERRORS
DDLREPLICATION	PACKAGE BODY	DBA_ERRORS
GGS_DDL_RULES	TABLE	DBA_TABLES
GGS_DDL_RULES_LOG	TABLE	DBA_TABLES
DDLAUX	PACKAGE	DBA_ERRORS
DDLAUX	PACKAGE BODY	DBA_ERRORS
DDLCTXINFO	PACKAGE	DBA_ERRORS
DDLCTXINFO	PACKAGE BODY	DBA_ERRORS
GGS_DDL_HIST	TABLE	DBA_TABLES
GGS_DDL_HIST_ALT	TABLE	DBA_TABLES
GGS_DDL_OBJECTS	TABLE	DBA_TABLES
GGS_DDL_COLUMNS	TABLE	DBA_TABLES
GGS_DDL_LOG_GROUPS	TABLE	DBA_TABLES
GGS_DDL_PARTITIONS	TABLE	DBA_TABLES
GGS_DDL_PRIMARY_KEYS	TABLE	DBA_TABLES
GGS_TEMP_COLS	TABLE	DBA_TABLES
GGS_TEMP_UK	TABLE	DBA_TABLES
GGS_DDL_SEQ	SEQUENCE	DBA_SEQUENCES
GGS_DDL_TRIGGER_BEFORE	TRIGGER	DBA_TRIGGERS

The script also checks for the ENC$ table which is created as part of the data dictionary in Oracle 11.2.

FILTERDDL

This function is declared in ddl_filter.sql

This function determines whether DDL for a specific object should be included or excluded.

It takes as input parameters the statement, owner, object name, object type and operation type and returns either INCLUDE or EXCLUDE.

The function can be customized, but after it has been modified, Oracle will no longer support it.

FILE_SEPARATOR

This function determines the file separator for the operating system. It will be '/' for Unix/Linux and '\\' for Windows.

The file separator is determined by selecting the value of USER_DUMP_DEST from V$PARAMETER and checking for the existence of '/' or '\\'.

Directories

GGS_DDL_TRACE

This directory is owned by the GGSCHEMA user. It defaults to the USER_DUMP_DEST directory.

For example for the single-instance NORTH database the GGS_DDL_TRACE directory is

```
/u01/app/oracle/diag/rdbms/north/NORTH/trace
```

Within the trace directory GoldenGate maintains a trace file recording all DDL activity. This file is called ggs_ddl_trace.log

Triggers

GGS_DDL_TRIGGER_BEFORE

This DDL trigger is owned by SYS. It is called every time a DDL statement is executed in the source database.

The trigger body is 1181 lines in size. The trigger makes numerous calls to procedures and functions in the DDLReplication package. It also inserts rows directly into the GGS_MARKER table.