Oracle® Fusion Middleware

Using Oracle GoldenGate for Heterogeneous Databases

18c (18.1.0)
E95982-04
October 2018

Oracle Fusion Middleware Using Oracle GoldenGate for Heterogeneous Databases, 18c (18.1.0)

E95982-04

Primary Author: Oracle Corp.

Contents

Preface

This guide helps you get started with using Oracle GoldenGate on heterogeneous database systems supported with this release.

Topics:

- Audience
- Documentation Accessibility
- Related Information
- Conventions

Audience

Using Oracle GoldenGate for Heterogeneous Databases is intended for DBA and system administrators who are responsible for implementing Oracle GoldenGate and managing the databases for an organization.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Information

The Oracle GoldenGate Product Documentation Libraries are found at

https://docs.oracle.com/en/middleware/goldengate/index.html

Additional Oracle GoldenGate information, including best practices, articles, and solutions, is found at:

Oracle GoldenGate A-Team Chronicles

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, such as "From the File menu, select **Save**." Boldface also is used for terms defined in text or in the glossary.
italic `italic`	Italic type indicates placeholder variables for which you supply particular values, such as in the parameter statement: `TABLE` *`table_name`*. Italic type also is used for book titles and emphasis.
`monospace` `MONOSPACE`	Monospace type indicates code components such as user exits and scripts; the names of files and database objects; URL paths; and input and output text that appears on the screen. Uppercase monospace type is generally used to represent the names of Oracle GoldenGate parameters, commands, and user-configurable functions, as well as SQL commands and keywords.
UPPERCASE	Uppercase in the regular text font indicates the name of a utility unless the name is intended to be a specific case.
{ }	Braces within syntax enclose a set of options that are separated by pipe symbols, one of which must be selected, for example: {*`option1`* | *`option2`* | *`option3`*}.
[ ]	Brackets within syntax indicate an optional element. For example in this syntax, the `SAVE` clause is optional: `CLEANUP REPLICAT` *`group_name`* `[, SAVE` *`count`*`]`. Multiple options within an optional element are separated by a pipe symbol, for example: `[`*`option1`* `|` *`option2`*`]`.

Part I

What is Oracle GoldenGate for Heterogeneous Databases?

Oracle GoldenGate is a comprehensive software package for real-time data capture and replication in heterogeneous IT environments.

The product set enables high availability solutions, real-time data integration, transactional change data capture, data replication, transformations, and verification between operational and analytical enterprise systems. Oracle GoldenGate 18c brings extreme performance with simplified configuration and management, support for cloud environments, expanded heterogeneity, and enhanced security.

You can use the following supported heterogeneous databases with Oracle GoldenGate.

- DB2 LUW
-
- DB2 for z/OS
- MySQL
- Teradata

Note:

Oracle GoldenGate 18c (18.1.0) is not released for SQL Server and DB2 for i. However, the documentation may include information associated with these databases.

Each database that Oracle GoldenGate supports has it’s own requirements and configuration. This book is divided into parts so that you can easily find information that is relevant to your environment. See *Installing Oracle GoldenGate* for system requirements and installation details for each of these databases.

Part II
Using Oracle GoldenGate for DB2 LUW Databases

With Oracle GoldenGate for DB2 LUW, you can perform initial loads and capture transactional data from supported DB2 LUW database versions and replicate the data to a DB2 LUW database or other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for DB2 LUW supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

This part describes tasks for configuring and running Oracle GoldenGate for DB2 LUW.

1 Understanding What's Supported for DB2 LUW

This chapter contains information on database and table features supported by Oracle GoldenGate for DB2 LUW.

Topics:

1.1 Supported DB2 LUW Data Types

Oracle GoldenGate supports all DB2 LUW data types, except those listed in Non-Supported DB2 LUW Data Types.

Limitations of Support

Oracle GoldenGate has the following limitations for supporting DB2 LUW data types:

- Oracle GoldenGate supports multi-byte character data types and multi-byte data stored in character columns. Multi-byte data is only supported in a like-to-like configuration. Transformation, filtering, and other types of manipulation are not supported for multi-byte character data.
- `BLOB` and `CLOB` columns must have a `LOGGED` clause in their definitions.
- `GRAPHIC` and `VARGRAPHIC` columns must be in a database, where the character set is UTF16. Any other character set causes the Oracle GoldenGate to abend.
- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- Extract fully supports the capture and apply of `TIMESTAMP(0)` through `TIMESTAMP(9)`. Extract also captures `TIMESTAMP(10)` through `TIMESTAMP(12)`, but it truncates the data to nanoseconds (maximum of nine digits of fractional time) and issues a warning to the error log. Replicat truncates timestamp data from other sources to nanoseconds when applying it to `TIMESTAMP(10)` through `TIMESTAMP(12)` in a DB2 LUW target.
- Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the timezone,

conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.

- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects that are larger than 4K. You can use the full Oracle GoldenGate functionality for objects that are 4K or smaller.
- Replication of XML columns between source and target databases with the *same* character set is supported. If the source and target database character sets are different, then XML replication may fail with a database error because some characters may not be recognized (or valid) in the target database character set.

1.2 Non-Supported DB2 LUW Data Types

The non-supported DB2 LUW data types are:

- `XMLType`
- `DECFLOAT`
- User-defined types
- Negative dates

1.3 Supported Objects and Operations for DB2 LUW

Object and operations that are supported for DB2 LUW are:

- Oracle GoldenGate Extract supports cross-endian capture where the database and Oracle GoldenGate are running on different byte order servers. The byte order is detected automatically for DB2 LUW version 10.5 or higher. If the DB2 database auto-detection on the DB2 LUW 10.5 database is not required then you can override it by specifying the `TRANLOGOPTIONS MIXEDENDIAN [ON|OFF]` parameter. For DB2 LUW version 10.1, this parameter must be used in the Extract parameter file for cross-endian capture. See `TRANLOGOPTIONS` in *Reference for Oracle GoldenGate*.
- Oracle GoldenGate supports the maximum number of columns and column size per table that is supported by the database.
- `TRUNCATE TABLE` for DB2 LUW version 9.7 and later.
- Multi Dimensional Clustered Tables (MDC) for DB2 LUW 9.5 and later.
- Materialized Query Tables. Oracle GoldenGate does not replicate the MQT itself, but only the base tables. The target database automatically maintains the content of the MQT based on the changes that are applied to the base tables by Replicat.
- Tables with `ROW COMPRESSION`. In DB2 LUW version 10.1 and later, `COMPRESS YES STATIC` is supported and `COMPRESS YES ADAPTIVE` are supported.
- Extended row size feature is enabled by default. It is supported with a workaround using `FETCHCOLS`. For any column values that are `VARCHAR` or `VARGRAPHIC` data types and are stored out of row in the database, you must fetch these extended rows by specifying these columns using the `FETCHCOLS` option in the `TABLE` parameter in the extract parameter file. With this option set, when the column values are out of row then Oracle GoldenGate will fetch its value. If the value is out of and `FETCHCOLS` is *not* specified then Extract will abend to prevent any data loss.

If you do not want to use this feature, set the `extended_row_size` parameter to `DISABLE`.

- Temporal tables with DB2 LUW 10.1 FixPack 2 and greater are supported. This is the default for Replicat.
- Limitations on Automatic Heartbeat Table support are as follows:
  - Oracle GoldenGate heartbeat parameters frequency and purge frequency are accepted in seconds and days. However, the DB2 LUW task scheduler accepts its schedule only in cron format so the Oracle GoldenGate input value to cron format may result in some loss of accuracy. For example:

    ```
    ADD HEARTBEATTABLE, FREQUENCY 150, PURGE_FREQUENCY 20
    ```

    This example sets the `FREQUENCY` to 150 seconds, which is converted to the closest minute value of 2 minutes, so the heartbeat table is updated every 120 seconds instead of every 150 seconds. Setting `PURGE_FREQUENCY` to 20 means that the history table is purged at midnight on every 20th day.
  - The following are steps are necessary for the heartbeat scheduled tasks to run:
    1. Set the `DB2_ATS_ENABLE` registry variable to `db2set DB2_ATS_ENABLE=YES`.
    2. Create the `SYSTOOLSPACE` tablespace if it does not already exist:

       ```
       CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP MANAGED BY AUTOMATIC
       STORAGE
       EXTENTSIZE 4
       ```
    3. Ensure instance owner has Database administration authority (DBADM):

       ```
       GRANT DBADM ON DATABASE TO instance_owner_name
       ```

1.4 Non-Supported Objects and Operations for DB2 LUW

Objects and operations for DB2 LUW that are not supported by Oracle GoldenGate are:

- Schema, table or column names that have trailing spaces
- Multiple instances of a database
- Datalinks
- Extraction or replication of DDL (data definition language) operations
- Generated columns (`GENERATE ALWAYS` clause)

Note:

1.5 Supported Object Names

For a list of characters that are supported in object names, see Supported Database Object Names in *Administering Oracle GoldenGate*.

2
Preparing the System for Oracle GoldenGate

This chapter describes how to prepare the environment to run Oracle GoldenGate on DB2 LUW.
Topics:

- Configuring the Transaction Logs for Oracle GoldenGate
- Preparing Tables for Processing
- Setting the Session Character Set
- Preparing for Initial Extraction
- Specifying the DB2 LUW Database in Parameter Files

2.1 Configuring the Transaction Logs for Oracle GoldenGate

To capture DML operations, Oracle GoldenGatereads the DB2 LUW online logs by default. However, it reads the archived logs if an online log is not available. To ensure the continuity and integrity of Oracle GoldenGateprocessing, configure the logs as follows.

- Retaining the Transaction Logs
- Specifying the Archive Path

2.1.1 Retaining the Transaction Logs

Configure the database to retain the transaction logs for roll forward recovery by enabling one of the following parameter sets, depending on the database version.

- DB2 LUW 9.5 and later:

  Set the `LOGARCHMETH` parameters as follows:

  – Set `LOGARCHMETH1` to `LOGRETAIN`.

  – Set `LOGARCHMETH2` to `OFF`.

  Alternatively, you can use any other `LOGARCHMETH` options, as long as forward recovery is enabled. For example, the following is valid:

  – Set `LOGARCHMETH1` to `DISK`.

  – Set `LOGARCHMETH2` to `TSM`.

To determine the log retention parameters:

1. Connect to the database.

   ```
   db2 connect to database user username using password
   ```

2. Get the database name.

```
db2 list db directory
```

3. Get the database configuration for the database.

   ```
   db2 get db cfg for database
   ```

   The fields to view are:

   ```
   Log retain for recovery status = RECOVERY
   User exit for logging status = YES
   ```

To set the log retention parameters:

1. Issue one of the following commands.

   To enable USEREXIT:

   ```
   db2 update db cfg for database using USEREXIT ON
   ```

   If not using USEREXIT, use this command:

   ```
   db2 update db cfg for database using LOGRETAIN RECOVERY
   ```

   To set LOGARCHMETH:

   ```
   db2 update db cfg for database using LOGARCHMETH1 LOGRETAIN
   db2 update db cfg for database using LOGARCHMETH2 OFF
   ```

2. Make a full backup of the database by issuing the following command.

   ```
   db2 backup db database to device
   ```

3. Place the backup in a directory to which DB2 LUW has access rights. If you get the following message, contact your systems administrator:

   ```
   SQL2061N An attempt to access media "device" is denied.
   ```

2.1.2 Specifying the Archive Path

Set the DB2 LUW OVERFLOWLOGPATH parameter to the archive log directory. The node attaches automatically to the path variable that you specify.

```
db2 connect to database
db2 update db cfg using overflowlogpath "path"
```

Exclude the node itself from the path. For example, if the full path to the archive log directory is /sdb2logarch/oltpods1/archive/OLTPODS1/NODE0000, then the OVERFLOWLOGPATH value should be specified as /sdb2logarch/oltpods1/archive/OLTPODS1.

2.2 Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

- Disabling Triggers and Cascade Constraints
- Assigning Row Identifiers
- Preventing Key Changes
- Enabling Change Capture

- Maintaining Materialized Query Tables

2.2.1 Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

2.2.2 Assigning Row Identifiers

Oracle GoldenGate requires some form of unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Using KEYCOLS to Specify a Custom Key

2.2.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

Note:

If there are other, non-usable keys on a table or if there are no keys at all on the table, Oracle GoldenGate logs an appropriate message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient `WHERE` clause.

2.2.2.2 Using `KEYCOLS` to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds.

2.2.3 Preventing Key Changes

Do not add columns to a key after Oracle GoldenGate starts extracting data from the table. This rule applies to a primary key, a unique key, a `KEYCOLS` key, or an all-column key. DB2 LUW does not supply a before image for columns that are added to a table. If any columns in a key are updated on the source, Oracle GoldenGate needs a before image to compare with the current values in the target table when it replicates the update.

2.2.4 Enabling Change Capture

Configure DB2 to log data changes in the expanded format that is supplied by the `DATA CAPTURE CHANGES` feature of the `CREATE TABLE` and `ALTER TABLE` commands. This format provides Oracle GoldenGate with the entire before and after images of rows that are changed by update statements. You can use GGSCI to issue the `ALTER TABLE` command as follows.

To Enable Change Capture from GGSCI:

1. From the Oracle GoldenGate directory, run GGSCI.
2. Log on to DB2 from GGSCI as a user that has `ALTER TABLE` privileges. Specify the data source name with `SOURCEDB` and specify the user login with `USERID` and `PASSWORD`.

   ```
   DBLOGIN SOURCEDB dsn, USERID user[, PASSWORD password]
   ```

3. Issue the following command. where *`owner.table`* is the fully qualified name of the table. You can use a wildcard to specify multiple table names. Only the asterisk (*) wildcard is supported for DB2 LUW.

   ```
   ADD TRANDATA owner.table
   ```

   `ADD TRANDATA` issues the following command, which includes logging the before image of `LONGVAR` columns:

   ```
   ALTER TABLE name DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;
   ```

Example 2-1 To Exclude `LONGVAR` Logging:

To omit the `INCLUDE LONGVAR COLUMNS` clause from the `ALTER TABLE` command, use `ADD TRANDATA` with the `EXCLUDELONG` option.

```
ADD TRANDATA owner.table, EXCLUDELONG
```

Note:

If `LONGVAR` columns are excluded from logging, the Oracle GoldenGate features that require before images, such as the `GETUPDATEBEFORES`, `NOCOMPRESSUPDATES`, and `NOCOMPRESSDELETES` parameters, might return errors if tables contain those columns. For a workaround, see the `REQUIRELONGDATACAPTURECHANGES | NOREQUIRELONGDATACAPTURECHANGES` options of the `TRANLOGOPTIONS` parameter.

2.2.5 Maintaining Materialized Query Tables

To maintain parity between source and target materialized query tables (MQT), you replicate the base tables, but not the MQTs. The target database maintains the MQTs based on the changes that Replicat applies to the base tables.

The following are the rules for configuring these tables:

- Include the base tables in your `TABLE` and `MAP` statements.
- Do not include MQTs in the `TABLE` and `MAP` statements.
- Wildcards can be used in `TABLE` and `MAP` statements, even though they might resolve MQT names along with regular table names. Oracle GoldenGate automatically excludes MQTs from wildcarded table lists. However, any MQT that is explicitly listed in an Extract `TABLE` statement by name will cause Extract to abend.

2.3 Setting the Session Character Set

To support the conversion of character sets between the source and target databases, make certain that the session character set is the same as the database character set. You can set the session character set with the `DB2CODEPAGE` environment variable.

2.4 Preparing for Initial Extraction

During the initialization of the Oracle GoldenGate environment, you will be doing an initial data synchronization and starting the Oracle GoldenGate processes for the first time. In conjunction with those procedures, you will be creating process groups. To create an Extract group, an initial start position must be established in the transaction log. This initial read position is on a transaction boundary that is based on one of the following:

- End of the transaction file
- A specific LRI value

The start point is specified with the `BEGIN` option of the `ADD EXTRACT` command.

When the Extract process starts for the first time, it captures all the transaction data that it encounters after the specified start point, but none of the data that occurred *before* that point. This can cause partial transactions to be captured if open transactions span the start point.

To ensure initial transactional consistency:

To avoid the capture of partial transactions, initialize the Extract process at a point in time when the database is in a paused state. DB2 LUW provides a `QUIESCE` command for such a purpose. This is the only way to ensure transactional consistency.

Note:

After the Extract is past the initialization, subsequent restarts of the Extract do not extract partial transactions, because the process uses recovery checkpoints to mark its last read position.

To view open transactions:

IBM provides a utility called `db2pd` for monitoring DB2 databases and instances. You can use it to view information about open transactions and to determine if any of them span the start point. However, because DB2 LUW log records lack timestamps, it might not be possible to make an accurate assessment. If possible, quiesce the database prior to initialization of Oracle GoldenGate.

For more information on initializing the Oracle GoldenGate environment, see Instantiating Oracle GoldenGate with an Initial Load in *Administering Oracle GoldenGate*.

2.5 Specifying the DB2 LUW Database in Parameter Files

For an Oracle GoldenGate process to connect to the correct DB2 LUW database, you must specify the name (not an alias) of the DB2 LUW database with the following parameters:

- Specify the DB2 source database with the Extract parameter `SOURCEDB`.
- Specify the DB2 target database name with the Replicat parameter `TARGETDB`.

For more information about these parameters, see the Reference for Oracle GoldenGate for Windows and UNIX.

3 Configuring Oracle GoldenGate for DB2 LUW

This chapter provides an overview of the basic steps required to configure Oracle GoldenGate for a DB2 LUW source and target database.
Topics:

3.1 What to Expect from these Instructions

These instructions show you how to configure basic parameter (configuration) files for the following processes:

- A primary Extract (captures transaction data from the data source)
- A data-pump Extract (propagates the data from local storage to the target system)
- A Replicat (applies replicated data to the target database)

Your business requirements probably will require a more complex topology, but this procedure forms a basis for the rest of your configuration steps.

By performing these steps, you can:

- Get the basic configuration files established.
- Build upon them later by adding more parameters as you make decisions about features or requirements that apply to your environment.
- Use copies of them to make the creation of additional parameter files faster than starting from scratch.

3.2 Where to Get More Information

See *Administering Oracle GoldenGate* and *Securing the Oracle GoldenGate Environment* for more information about:

- The processes and files that you are configuring
- Detailed configuration information

- Security options
- Data-integration options (filtering, mapping, conversion)
- Instructions for configuring complex topologies
- Steps to perform initial instantiation of the replication environment

3.3 Configuring the Primary Extract

These steps configure the primary Extract to capture transaction data from a source DB2 LUW and write the data to a local trail for temporary storage.

1. In GGSCI on the source system, create the Extract parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: `name` is the name of the primary Extract.

2. Enter the Extract parameters in the order shown, starting a new line for each parameter statement.

   Basic parameters for the primary Extract

   ```
   EXTRACT finance
   SOURCEDB mysource, USERIDALIAS myalias
   ENCRYPTTRAIL AES192
   EXTTRAIL /ggs/dirdat/lt
   TABLE hr.*;
   ```

Parameter	Description
`EXTRACT` *`group`*	*`group`* is the name of the Extract group.
`SOURCEDB` *`database`*`,` `USERIDALIAS` *`alias`*	Specifies the real name of the source DB2 for i database (not an alias), plus the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store. For more information, see Database User for Oracle GoldenGate Processes.
`ENCRYPTTRAIL` *`algorithm`*	Encrypts the local trail.
`EXTTRAIL` *`pathname`*	Specifies the path name of the local trail to which the primary Extract writes captured data for temporary storage.
`TABLE` *`schema.object`*`;`	Specifies the database object for which to capture data. • `TABLE` is a required keyword. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the table name, or a wildcarded set of tables. The question mark (?) wildcard is not supported for this database. Note that only the asterisk (*) wildcard is supported for DB2 LUW. Terminate the parameter statement with a semi-colon. To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter. For more information and for additional options that control data filtering, mapping, and manipulation, see `TABLE | MAP` in *Reference for Oracle GoldenGate*.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI.
4. Save and close the file.

3.4 Configuring the Data Pump Extract

These steps configure the data pump that reads the local trail and sends the data across the network to a remote trail on the target. The data pump is optional, but recommended.

1. In GGSCI on the source system, create the data-pump parameter file.

   ```
   EDIT PARAMS name
   ```

   Where *name* is the name of the data-pump Extract.
2. Enter the data-pump Extract parameters in the order shown, starting a new line for each parameter statement. Your input variables will be different.

   Basic parameters for the data-pump Extract group:

   ```
   EXTRACT extpump
   SOURCEDB mypump, USERIDALIAS myalias
   RMTHOST fin1, MGRPORT 7809 ENCRYPT AES192, KEYNAME securekey2
   RMTTRAIL /ggs/dirdat/rt
   TABLE hr.*;
   ```

Parameter	Description
`EXTRACT` *`group`*	*`group`* is the name of the data pump Extract.
`SOURCEDB` *`database`*`,` `USERIDALIAS` *`alias`*	Specifies the real name of the source DB2 LUW database (not an alias), plus the alias of the database login credential of the user that is assigned to Extract. This credential must exist in the Oracle GoldenGate credential store.
`RMTHOST` *`hostname`*`,` `MGRPORT` *`portnumber`*`,` `[, ENCRYPT` *`algorithm`* `KEYNAME` *`keyname`*`]`	• `RMTHOST` specifies the name or IP address of the target system. • `MGRPORT` specifies the port number where Manager is running on the target. • `ENCRYPT` specifies optional encryption of data across TCP/IP.
`RMTTRAIL` *`pathname`*	Specifies the path name of the remote trail.
`TABLE` *`schema.object`*`;`	Specifies a table or sequence, or multiple objects specified with a wildcard. In most cases, this listing will be the same as that in the primary Extract parameter file. • `TABLE` is a required keyword. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the name of a table or a wildcarded set of tables. Only the asterisk (*) wildcard is supported for DB2 LUW. The question mark (?) wildcard is not supported for this database. Terminate the parameter statement with a semi-colon. To exclude tables from a wildcard specification, use the `TABLEEXCLUDE` parameter. For more information and for additional `TABLE` options that control data filtering, mapping, and manipulation, see `TABLE | MAP` in *Reference for Oracle GoldenGate*.

3. Enter any optional Extract parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI.
4. Save and close the file.

3.5 Configuring Replicat

These steps configure the Replicat process in a basic way without any special mapping or conversion of the data.

1. In GGSCI on the target system, create the Replicat parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: *name* is the name of the Replicat group.
2. Enter the Replicat parameters in the order shown, starting a new line for each parameter statement.

   ```
   REPLICAT financer
   TARGETDB FINANCIAL USERID ogg, PASSWORD AACAAAAAAAAAAA, BLOWFISH ENCRYPTKEY mykey
   ASSUMETARGETDEFS
   -- Instead of ASSUMETARGETDEFS, use SOURCEDEFS if replicating from
   -- DB2 LUW to a different database type, or from a DB2 DB2 LUW source
   -- that is not identical in definitions to a target DB2 LUW database.
   -- SOURCEDEFS /users/ogg/dirdef/defsfile
   DISCARDFILE /users/ogg/disc
   MAP hr.*, TARGET hr2.*;
   ```

Parameter	Description
`REPLICAT` *group*	*group* is the name of the Replicat group.
`TARGETDB` *database* `USERID` *user*, `PASSWORD` *password*, `BLOWFISH ENCRYPTKEY` *keyname*	Specifies database connection information. • `SOURCEDB` specifies the data source name (DSN) of the target DB2 LUW database. • `USERID` specifies the Replicat database user profile. • `PASSWORD` specifies the user's password that was encrypted with the `ENCRYPT PASSWORD` command. Enter or paste the encrypted password after the `PASSWORD` keyword. • `BLOWFISH ENCRYPTKEY` *keyname* specifies the name of the lookup key in the local `ENCKEYS` file.
`DECRYPTTRAIL BLOWFISH`	Decrypts the input trail.
`SOURCEDEFS` *pathname* | `ASSUMETARGETDEFS`	Specifies how to interpret data definitions. Use `SOURCEDEFS` if the source and target tables have different definitions, such as when replicating data between dissimilar IBM databases or from an IBM database to an Oracle database. For *pathname*, specify the source data-definitions file that you created with the `DEFGEN` utility. Use `ASSUMETARGETDEFS` if the source and target tables are all DB2 LUW and have the same definitions.

Parameter	Description
`MAP` *`owner.table`*`,` `TARGET` *`owner.table`*`;`	Specifies a relationship between a source and target table or tables. The `MAP` clause specifies the source objects, and the `TARGET` clause specifies the target objects to which the source objects are mapped. • *`owner`* is the schema or library name. • *`table`* is the name of a table or a wildcard definition for multiple tables. Terminate the `MAP` statement with a semi-colon. To exclude tables from a wildcard specification, use the `MAPEXCLUDE` parameter. For more information and for additional options that control data filtering, mapping, and manipulation, see `MAP` in *Reference for Oracle GoldenGate*.

3. Enter any optional Extract parameters that are recommended elsewhere in this manual and any others shown in Summary of Extract Parameters.
4. Save and close the file.

- Creating a Temporal Table
- Creating a Checkpoint Table
- Configuring the Replicat Parameter File

3.5.1 Creating a Temporal Table

A temporal table is a table that maintains the history of its data and the time period when its data are valid. Temporal tables are used in Oracle GoldenGate to keep track of all the old rows that are deleted or updated in the table. Temporal tables are also used to maintain the business validity of its rows and data. For example, Oracle GoldenGate keeps track of the time period during which a row is valid. There are three types of temporal tables, system-period, application-period, and bitemporal table.

- Support for Temporal Tables
- Replicating with Temporal Tables
- Converting

3.5.1.1 Support for Temporal Tables

- Replication between system-period temporal tables and application-period temporal tables is not supported.
- Replication from a non-temporal table to a temporal table is not supported.
- Replication of temporal tables with the `INSERTALLRECORDS` parameter is not supported.
- Bidirectional replication is supported only with the default replication.
- CDR in bidirectional replication is not supported.
- CDR in application-period temporal tables is supported.

3.5.1.2 Replicating with Temporal Tables

You can choose one of the following methods to replicate a system-period or a bitemporal temporal table as follows:

- You can replicate a temporal table to another temporal table only; this is the default behavior. Oracle GoldenGate will not replicate the `SYSTEM_TIME` period and transaction id columns because these are automatically generated columns at the apply side. The database manager populates the columns in the target temporal table using the system clock time and with the default values. You can preserve the original values these columns then use any of the following:
  – Add extra timestamp columns in the target temporal table and map the columns accordingly. The extra columns are automatically added to the associated history table.
  – Use a non-temporal table at the apply side and map the columns appropriately. In this scenario, you will not be able to maintain the history table.
  – In a heterogeneous configuration where the source is DB2 LUW and the target is a different database, you can either ignore the automatically generated columns or use an appropriate column conversion function to convert the columns value in the format that target database supports and map them to target columns accordingly.

  Or
- You can replicate a temporal table, with the associated history table, to a temporal and history table respectively then you must specify the replicate parameter, `DBOPTIONS SUPPRESSTEMPORALUPDATES`. You must specify both the temporal table and history table to be captured in the Extract parameter file. Oracle GoldenGate replicates the `SYSTEM_TIME` period and transactions id columns value. You must ensure that the database instance has the execute permission to run the stored procedure at the apply side.

Oracle GoldenGate cannot detect and resolve conflicts while using default replication as `SYSTEM_TIME` period and `transactionstart id` columns remains auto generated. These columns cannot be specified in `set` and `where` clause. If you use the `SUPPRESSTEMPORALUPDATES` parameter, then Oracle GoldenGate supports CDR.

3.5.1.3 Converting

You can convert an already existing table into a temporal table, which changes the structure of the table. This section describes how the structure of the tables changes. The following sample existing table is converted into all three temporal tables types in the examples in this section:.

```
Table policy_info
(
Policy_id char[4] not null primary key,
Coverage int not null
              )
And the tables contains the following initial rows
                   POLICY_ID       COVERAGE
                   -------------        -----------
                     ABC                12000
                     DEF                 13000
                     ERT                 14000
```

Example 1 Converting an existing table into System-period temporal table.
You convert the sample existing table into a system-period temporal table by adding `SYSTEM_PERIOD`, `transaction id` columns, and `SYSTEM_TIME` period as in the following:

```
ALTER TABLE policy_info
   ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE policy_info
   ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE policy_info
   ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE policy_info ADD PERIOD SYSTEM_TIME(sys_start, sys_end);
```

Then you create a history table for the new temporal table using one of the following two methods:

-
  ```
      CREATE TABLE hist_policy_info
  (
   policy_id     CHAR(4) NOT NULL,
  coverage     INT NOT NULL,
  sys_start    TIMESTAMP(12) NOT NULL ,
  sys_end      TIMESTAMP(12) NOT NULL,
  ts_id            TIMESTAMP(12) NOT NULL
              );
   ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
  ```

-
  ```
  CREATE TABLE hist_policy_info LIKE policy_info with RESTRICT ON DROP;
  ```

  The `RESTRICT ON DROP` clause will not allow the history table to get dropped while dropping system-period temporal table. Otherwise the history table gets implicitly dropped while dropping its associated temporal table. You can create a history table without `RESTRICT ON DROP.` A history table cannot be explicitly dropped.

  You should not use the `GENERATED ALWAYS` clause while creating a history table. The primary key of the system-period temporal table also does not apply here as there could be many updates for a particular row in the base table, which triggers many inserts into the history table for the same set of primary keys. Apart from these, the structure of a history table should be exactly same as its associated system-period temporal table. The history table must have the same number and order of columns as system-period temporal table. History table columns cannot explicitly be added, dropped, or changed. You must associate a system-period temporal table with its history table with the following statement:

  ```
   ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info.
  ```

  The `GENERATED ALWAYS` columns of the table are the ones that are always populated by the database manager so you do not have any control over these columns. The database manager populates these columns based on the system time.

  The extra added `SYSTEM_PERIOD` and `transaction id` columns will have default values for already existing rows as in the following:

  ```
  POLICY_ID                                   COVERAGE
  SYS_START
  SYS_END                                                TS_ID
  --------- ----------- --------------------------------
  ```

```
-------------------------------
-----------------------------------------------------------------------------
ABC             12000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
DEF             13000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
ERT             14000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
```

The associated history table is populated with the before images once you start updating the temporal table.

Example 2 Converting an existing table into application-period temporal table.
You can convert the sample existing table into application-period temporal table by adding time columns and a `BUSINESS_TIME` period as in the following:

```
ALTER TABLE policy_info ADD COLUMN bus_start DATE NOT NULL DEFAULT '10/10/2001'"
ALTER TABLE policy_info ADD COLUMN bus_end DATE NOT NULL DEFAULT '10/10/2002'
ALTER TABLE policy_info ADD PERIOD BUSINESS_TIME(bus_start, bus_end)
```

While adding time columns, you need to make sure that while entering business validity time values of the existing time columns, the `bus_start` column always has value lesser than `bus_end` because these columns specify the business validity of the rows.
The new application-period temporal table will look similar to:

```
POLICY_ID   COVERAGE    BUS_START  BUS_END
--------- ----------- ---------- ------------------------------
ERT             14000             10/10/2001  10/10/2002
DEF             13000            10/10/2001   10/10/2002
ABC             12000            10/10/2001   10/10/2002
```

Example 3 Converting an existing table into bitemporal table.
You can convert the sample existing table into bitemporal table by adding `SYSTEM_PERIOD`, time columns along with the `SYSTEM_TIME` and `BUSINESS_TIME` period as in the following:

```
ALTER TABLE policy_info
   ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE policy_info
   ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE policy_info
   ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE policy_info ADD PERIOD SYSTEM_TIME(sys_start, sys_end);

ALTER TABLE policy_info ADD COLUMN bus_start DATE NOT NULL DEFAULT '10/10/2001'"
ALTER TABLE policy_info ADD COLUMN bus_end DATE NOT NULL DEFAULT '10/10/2002'
ALTER TABLE policy_info ADD PERIOD BUSINESS_TIME(bus_start, bus_end)
```

While adding the time columns, you must make sure that while entering business validity time values of already existing time columns, the `bus_start` column always has value lesser than `bus_end` because these columns specify the business validity of the rows.
Then you create a history table for the new temporal table using one of the following two methods:

-
  ```
      CREATE TABLE hist_policy_info
  (
   policy_id     CHAR(4) NOT NULL,
  coverage     INT NOT NULL,
  sys_start    TIMESTAMP(12) NOT NULL ,
  sys_end      TIMESTAMP(12) NOT NULL,
  ts_id            TIMESTAMP(12) NOT NULL
              );
   ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
  CREATE TABLE hist_policy_info LIKE policy_info with RESTRICT ON DROP;
  ```

- The `RESTRICT ON DROP` clause will not allow the history table to get dropped while dropping system-period temporal table. Otherwise the history table gets implicitly dropped while dropping its associated temporal table. You can create a history table without `RESTRICT ON DROP.` A history table cannot be explicitly dropped.

  You should not use the `GENERATED ALWAYS` clause while creating a history table. The primary key of the system-period temporal table also does not apply here as there could be many updates for a particular row in the base table, which triggers many inserts into the history table for the same set of primary keys. Apart from these, the structure of a history table should be exactly same as its associated system-period temporal table. The history table must have the same number and order of columns as system-period temporal table. History table columns cannot explicitly be added, dropped, or changed. You must associate a system-period temporal table with its history table with the following statement:

  ```
   ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info.
  ```

  The `GENERATED ALWAYS` columns of the table are the ones that are always populated by the database manager so you do not have any control over these columns. The database manager populates these columns based on the system time.

  The extra added `SYSTEM_PERIOD` and `transaction id` columns will have default values for already existing rows as in the following:

  ```
  POLICY_ID                                   COVERAGE
  SYS_START
  SYS_END                                                TS_ID
  --------- ----------- -------------------------------
  -------------------------------
  -------------------------------------------------------------------------------
  ABC             12000 0001-01-01-00.00.00.000000000000
  9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
  DEF             13000 0001-01-01-00.00.00.000000000000
  9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
  ERT             14000 0001-01-01-00.00.00.000000000000
  9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
  ```

  The associated history table is populated with the before images once you start updating the temporal table.

The extra added `SYSTEM_TIME` period, transaction id and time columns will have default values for already existing rows as in the following:

```
POLICY_ID COVERAGE    SYS_START
SYS_END                          TS_ID                             BUS_START
BUS_END
--------- ----------- -------------------------------
```

```
------------------------------ ------------------------------ ----------
-----------------------------------
ABC   12000 0001-01-01-00.00.00.000000000000 9999-12-30-00.00.00.000000000000
0001-01-01-00.00.00.000000000000 10/10/2001 10/10/2002
DEF             13000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000 10/10/2001
10/10/2002
ERT             14000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000 10/10/2001
10/10/2002
```

The history table is populated with the before images once user starts updating the temporal table.

Example 4 Replication in Heterogeneous Environment.
In heterogeneous configuration in which you do not have temporal tables at the apply side, you can only replicate the system-period and bitemporal tables though *not* the associated history tables. While performing replication in this situation, you must take care of the `SYSTEM_PERIOD` and transaction id columns value. These columns will have some values that the target database might not support. You should first use the map conversion functions to convert these values into the format that the target database supports, and then map the columns accordingly.
For example, MySQL has a `DATETIME` range from `1000-01-01 00:00:00.000000` to `9999-12-31 23:59:59.999999`. You cannot replicate a timestamp value of `0001-01-01-00.00.00.000000000000` to MySQL. To replicate such values, you must convert this value into the MySQL `DATETIME` value `1000-01-01 00:00:00.000000`, and then map the columns. If you have the following row in the `policy_info` system-period table:

```
POLICY_ID                                   COVERAGE
SYS_START
SYS_END                                                TS_ID
--------- ----------- -------------------------------
-------------------------------
-------------------------------------------------------------------------------
ABC             12000 0001-01-01-00.00.00.000000000000
9999-12-30-00.00.00.000000000000 0001-01-01-00.00.00.000000000000
```

To replicate the row into MySQL, you would use the `colmap()` function:

```
map source_schema.policy_info, target target_schema.policy_info colmap
(policy_id=policy_id, coverage=coverage, sys_start= @IF( ( @NUMSTR( @STREXT(sys_
start,1,4))) > 1000, sys_start, '1000-01-01 00.00.00.000000'), sys_end=sys_end,
 ts_id= @IF( ( @NUMSTR( @STREXT(ts_id,1,4))) > 1000, ts_id, '1000-01-01
 00.00.00.000000'));
```

3.5.2 Creating a Checkpoint Table

The checkpoint table is a required component of Replicat.

Replicat maintains its recovery checkpoints in the checkpoint table, which is stored in the target database. Checkpoints are written to the checkpoint table within the Replicat transaction. Because a checkpoint either succeeds or fails with the transaction, Replicat ensures that a transaction is only applied once, even if there is a failure of the process or the database.

To configure a checkpoint table, see Creating a Checkpoint Table in *Administering Oracle GoldenGate*.

3.5.3 Configuring the Replicat Parameter File

These steps configure the Replicat process. This process applies replicated data to a DB2 LUW target database.

1. In GGSCI on the target system, create the Replicat parameter file.

   ```
   EDIT PARAMS name
   ```

   Where: `name` is the name of the Replicat group.

2. Enter the Replicat parameters in the order shown, starting a new line for each parameter statement.

   Basic parameters for the Replicat group:

   ```
   REPLICAT financer
   TARGETDB mytarget, USERIDALIAS myalias
   ASSUMETARGETDEFS
   MAP hr.*, TARGET hr2.*;
   ```

Parameter	Description
`REPLICAT` *`group`*	*`group`* is the name of the Replicat group.
`TARGETDB` *`database`*`,` `USERIDALIAS` *`alias`*	Specifies the real name of the target DB2 LUW database (not an alias), plus the alias of the database login credential of the user that is assigned to Replicat. This credential must exist in the Oracle GoldenGate credential store. For more information, see Database User for Oracle GoldenGate Processes.
`ASSUMETARGETDEFS`	Specifies how to interpret data definitions. `ASSUMETARGETDEFS` assumes the source and target tables have identical definitions. (This procedure assume identical definitions.) Use the alternative `SOURCEDEFS` if the source and target tables have different definitions, and create a source data-definitions file with the `DEFGEN` utility.
`MAP` *`schema.object`*`,` `TARGET` *`schema.object`*`;`	Specifies the relationship between a source table or multiple objects, and the corresponding target object or objects. • `MAP` specifies the source portion of the `MAP` statement and is a required keyword. Specify the source objects in this clause. • `TARGET` specifies the target portion of the `MAP` statement and is a required keyword. Specify the target objects to which you are mapping the source objects. • *`schema`* is the schema name or a wildcarded set of schemas. • *`object`* is the name of a table or a wildcarded set of objects. Terminate this parameter statement with a semi-colon. Note that only the asterisk (*) wildcard is supported for DB2 LUW. The question mark (?) wildcard is not supported for this database. To exclude objects from a wildcard specification, use the `MAPEXCLUDE` parameter.

3. Enter any optional Replicat parameters that are recommended for your configuration. You can edit this file at any point before starting processing by using the `EDIT PARAMS` command in GGSCI.

4. Save and close the file.

3.6 Next Steps in the Deployment

Because of its flexibility, Oracle GoldenGate offers numerous features and options that must be considered before you start any processes. To further configure Oracle GoldenGate to suit your business needs, see the following:

- For additional configuration guidelines to achieve specific replication topologies, see *Administering Oracle GoldenGate*. This guide also contains information about:
  - Oracle GoldenGate architecture
  - Oracle GoldenGate commands
  - Oracle GoldenGate initial load methods
  - Configuring security
  - Using customization features
  - Mapping columns that contain dissimilar data
  - Data filtering and manipulation
- For syntax options and descriptions of Oracle GoldenGate GGSCI commands and Oracle GoldenGate parameters shown in this guide, see *Reference for Oracle GoldenGate*.

3.7 When to Start Replicating Transactional Changes

You must start replication when the source and target data is in a synchronized state, where the corresponding rows in the source and target tables contain identical data values. Unless you are starting with brand new source and target databases with no current user activity, you will need to activate change capture and apply processes to handle ongoing transactional changes while an initial load is being applied to the target. This process is known as *initial synchronization*, or also as *instantiation*. The initial load captures a point-in-time snapshot of the source data and applies it to the target, while Oracle GoldenGate maintains any changes that are made after that point.

See Instantiating Oracle GoldenGate with an Initial Load in *Administering Oracle GoldenGate* for instantiation options.

3.8 Testing Your Configuration

It is important to test your configuration in a test environment before deploying it live on your production machines. This is especially important in an active-active or high availability configuration, where trusted source data may be touched by the replication processes. Testing enables you to find and resolve any configuration mistakes or data issues without the need to interrupt user activity for re-loads on the target or other troubleshooting activities.

Part III
Using Oracle GoldenGate for DB2 for z/OS

With Oracle GoldenGate for DB2 for z/OS, you can perform initial loads and capture transactional data from supported DB2 for z/OS versions and replicate the data to a DB2 for z/OS database or other supported Oracle GoldenGate targets, such as an Oracle Database.

Oracle GoldenGate for DB2 for z/OS is installed and runs remotely on Linux, zLinux, or AIX.

Oracle GoldenGate for DB2 for z/OS supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

Topics:

4

Understanding What's Supported for DB2 for z/OS

This chapter contains information on database and table features supported byOracle GoldenGate for DB2 z/OS.

Topics:

4.1 Supported DB2 for z/OS Data Types

This section lists the DB2 for z/OS data types that Oracle GoldenGate supports and any limitations of this support.

- Oracle GoldenGate does not perform character set conversion for columns that could contain multi-byte data. This includes `GRAPHIC`, `VARGRAPHIC` and `DBCLOB` data types, as well as `CHAR`, `VARCHAR`, and `CLOB` for tables defined with `ENCODING_SCHEME` of '`M`' (multiple CCSID set or multiple encoding schemes) or '`U`' (Unicode). Such data is only supported if the source and target systems are the same CCSID.
- Oracle GoldenGate supports ASCII, EBCDIC, and Unicode data format. Oracle GoldenGate converts between ASCII and EBCDIC data automatically. Unicode is not converted.
- Oracle GoldenGate supports most DB2 data types except those listed in Non-Supported DB2 for z/OS Data Types.

Limitations of Support

- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- Oracle GoldenGate does not support the filtering, column mapping, or manipulation of large objects greater than 4K in size. You can use the full Oracle GoldenGate functionality for objects that are 4K or smaller.
- Oracle GoldenGate supports the default `TIMESTAMP` and the `TIMESTAMP` with `TIMEZONE` to up to 9 digit fractional value, but no further.

4.2 Non-Supported DB2 for z/OS Data Types

This section lists DB2 for z/OS data types that Oracle GoldenGate does not support. Data that is not supported may affect the integrity of the target data in relation to the source data.

- `XML`
- User-defined types
- Negative dates

4.3 Supported Objects and Operations for DB2 for z/OS

This section lists the database objects and types of operations that Oracle GoldenGate supports.

- Extraction and replication of DML operations on DB2 for z/OS tables that contain rows of up to 512KB in length. This size exceeds the maximum row size of DB2.
- `INSERT` operations from the IBM `LOAD` utility are supported for change capture if the utility is run with `LOG YES` and `SHRLEVEL CHANGE`, and the source tables that are being loaded have `DATA CAPTURE CHANGES` enabled (required by Oracle GoldenGate) and are specified in the Oracle GoldenGate Extract configuration. Oracle GoldenGate also supports initial loads with the `LOAD` utility to instantiate target tables during initial synchronization.
- Oracle GoldenGate supports the maximum number of columns per table, which is supported by the database.
- Oracle GoldenGate supports the maximum column size that is supported by the database.
- Extraction and replication of data that is stored using DB2 data compression (`CREATE TABLESPACE COMPRESS YES`).
- `TRUNCATE TABLE` is supported, but because this command issues row deletes to perform the truncate, they are shown in Oracle GoldenGate statistics as such, and not as a truncate operation. To replicate a `TRUNCATE` , the Replicat process uses a `DELETE` operation without a `WHERE` clause.
- `TRUNCATES` are always captured from a DB2 for z/OS source, but can be ignored by Replicat if the `IGNORETRUNCATES` parameter is used in the Replicat parameter file.
- `UNICODE` columns in `EBCDIC` tables are supported.

4.4 Non-Supported Objects and Operations for DB2 for z/OS

The following objects and operations are not supported by Oracle GoldenGate on DB2 for z/OS:

- Extraction or replication of DDL operations
- Clone tables

- Data manipulation, including compression, that is performed within user-supplied DB2 exit routines, such as:
  - Date and time routines
  - Edit routines (`CREATE TABLE EDITPROC`)
  - Validation routines (`CREATE TABLE VALIDPROC`)
- Replicating with `BATCHSQL` is not fully functional for DB2 for z/OS. Non-insert operations are not supported so any update or delete operations will cause Replicat to drop temporarily out of `BATCHSQL` mode. The transactions will stop and errors will occur.

5

Preparing the DB2 for z/OS Database for Oracle GoldenGate

Learn how to prepare your database and environment to support Oracle GoldenGate.
Topics:

5.1 Preparing Tables for Processing

You must perform the following tasks to prepare your tables for use in an Oracle GoldenGate environment.

5.1.1 Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

- A delete is issued for `emp_src`.
- It cascades a delete to `salary_src`.
- Oracle GoldenGate sends both deletes to the target.
- The parent delete arrives first and is applied to `emp_targ`.
- The parent delete cascades a delete to `salary_targ`.
- The cascaded delete from `salary_src` is applied to `salary_targ`.
- The row cannot be located because it was already deleted in step 5.

5.1.2 Assigning Row Identifiers

Oracle GoldenGate requires some form of unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Using KEYCOLS to Specify a Custom Key

5.1.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

Note:

If there are other, non-usable keys on a table or if there are no keys at all on the table, Oracle GoldenGate logs an appropriate message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient `WHERE` clause.

5.1.2.2 Using `KEYCOLS` to Specify a Custom Key

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds. For more information, see *Reference for Oracle GoldenGate*.

5.1.3 Handling `ROWID` Columns

Any attempt to insert into a target table that includes a column with a data type of `ROWID GENERATED ALWAYS` (the default) will fail with the following ODBC error:

```
ODBC error: SQLSTATE 428C9 native database error -798. {DB2 FOR OS/390}{ODBC DRIVER}
{DSN08015} DSNT408I SQLCODE = -798, ERROR: YOU CANNOT INSERT A VALUE INTO A COLUMN
THAT IS DEFINED WITH THE OPTION GENERATED ALWAYS. COLUMN NAME ROWIDCOL.
```

You can do one of the following to prepare tables with `ROWID` columns to be processed by Oracle GoldenGate:

- Ensure that any `ROWID` columns in target tables are defined as `GENERATED BY DEFAULT`.
- If it is not possible to change the table definition, you can work around it with the following procedure.

To Work Around `ROWID GENERATE ALWAYS`:

1. For the source table, create an Extract `TABLE` statement, and use a `COLSEXCEPT` clause in that statement that excludes the `ROWID` column. For example:

   ```
   TABLE tab1, COLSEXCEPT (rowidcol);
   ```

   The `COLSEXCEPT` clause excludes the `ROWID` column from being captured and replicated to the target table.
2. For the target table, ensure that Replicat does not attempt to use the `ROWID` column as the key. This can be done in one of the following ways:
   - Specify a primary key in the target table definition.
   - If a key cannot be created, create a Replicat `MAP` parameter for the table, and use a `KEYCOLS` clause in that statement that contains any unique columns except for the `ROWID` column. Replicat will use those columns as a key. For example:

     ```
     MAP tab1, TARGET tab1, KEYCOLS (num, ckey);
     ```

For more information about `KEYCOLS`, see Assigning Row Identifiers.

5.2 Configuring a Database Connection

This section contains instructions for setting up the Extract and Replicat connections to a SQL Server database.

- Setting Initialization Parameters
- Specifying the Path to the Initialization File
- Ensuring ODBC Connection Compatibility
- Specifying the Number of Connection Threads

5.2.1 Setting Initialization Parameters

The following DB2 for z/OS initialization parameters apply to Oracle GoldenGate and must be set correctly before starting Oracle GoldenGate processes.

- `MVSDEFAULTSSID`: set to the DB2 subsystem.
- `LOCATION`: set to the DB2 location name as stored in the DB2 Boot Strap Dataset.
- `MVSATTACHTYPE`: set to `RRSAF` (Recoverable Resource Manager Services Attachment Facility) or `CAF` (Call Attachment Facility). IBM recommends using `RRSAF`.
- `MULTICONTEXT:` set to 1 if using `RRSAF`.
- `PLANNAME`: set to the DB2 plan. The default plan name is `DSNACLI`.

Do not use the `CURRENTAPPENSCH` initialization parameter (keyword).

Note:

When using the `CAF` attachment type, you must use the Oracle GoldenGate `DBOPTIONS` parameter with the `NOCATALOGCONNECT` option in the parameter file of any Extract or Replicat process that connects to DB2. This parameter disables the usual attempt by Oracle GoldenGate to obtain a second thread for the DB2 catalog. Otherwise, you will receive error messages, such as: `ODBC operation failed: Couldn't connect to` *`data source`* `for catalog queries.`

5.2.2 Specifying the Path to the Initialization File

Specify the ODBC initialization file by setting the `DSNAOINI` environment variable in the z/OS UNIX profile, as in the following example:

```
export DSNAOINI="/etc/odbc810.ini"
```

5.2.3 Ensuring ODBC Connection Compatibility

To ensure that you configure the DB2 ODBC initialization file correctly, follow the guidelines in the *DB2 UDB for z/OS ODBC Guide and Reference* manual. One important consideration is the coding of the open and close square brackets (the [character and the] character). The square bracket characters are "variant" characters that are encoded differently in different coded character set identifiers (CCSID), but must be of the IBM-1047 CCSID in the ODBC initialization file. DB2 ODBC does not recognize brackets of any other CCSID. Note the following:

- The first (or open) bracket must use the hexadecimal characters `X'AD'` (`0xAD`).
- The second (or close) bracket must use the hexadecimal characters `X'BD'` (`0xBD`).

To set the correct code for square brackets, use any of the following methods.

- Use the `hex` command in OEDIT and change the hex code for each character appropriately.
- Use the `iconv` utility to convert the ODBC initialization file. For example, to convert from CCSID IBM-037 to IBM-1047, use the following command:

  ```
  iconv -f IBM-037 -t IBM-1047 ODBC.ini > ODBC-1047.ini

  mv ODBC-1047.ini ODBC.ini
  ```
- Change your terminal emulator or terminal configuration to use CCSID IBM-1047 when you create or alter the file.

5.2.4 Specifying the Number of Connection Threads

Every Oracle GoldenGate process makes a database connection. Depending on the number of processes that you will be using and the number of other DB2 connections that you expect, you might need to adjust the following DB2 system parameters on the DSNTIPE DB2 Thread Management Panel:

- `MAX USERS` (macro `DSN6SYSP CTHREAD`)

- `MAX TSO CONNECT` (macro `DSN6SYSP IDFORE`)
- `MAX BATCH CONNECT` (macro `DSN6SYSP IDBACK`)

If using RRSAF, allow:

- Two DB2 threads per process for each of the following:
  - Extract
  - Replicat
  - The GGSCI command `DBLOGIN` (logs into the database)
  - `DEFGEN` utility (generates data definitions for column mapping)
- One extra DB2 thread for Extract for IFI calls.
- One extra DB2 thread for each `SQLEXEC` parameter statement that will be issued by each Extract and Replicat process. For more information about `SQLEXEC`, see the *Reference for Oracle GoldenGate*.

If using CAF, there can be only one thread per Oracle GoldenGate process.

5.3 Accessing Load Modules

Grant Oracle GoldenGate USS access to the `SDSNLOAD` system load library and to the `DSNHDECP` load module. You can include the libraries in one of the following places:

- The z/OS system search order.
- The USS profile of the Oracle GoldenGate user. Use a UNIX command similar to the following, where `DSN810` is the user-assigned data set prefix from the DB2 installation.

  ```
  export STEPLIB='DSN810.SDSNEXIT:DSN810.SDSNLOAD'
  ```

The preceding command will cause USS to allocate the equivalent of a `STEPLIB DD` statement whenever it executes a shell command or Oracle GoldenGate process. If using APF, all libraries in the `STEPLIB` concatenation must be APF-authorized.

5.4 Specifying Job Names and Owners

By default, USS sets the job name and owner of all Oracle GoldenGate processes to that of the user who started them. You can change the job name or user by setting the `_BPX_JOBNAME` and `_BPX_USERID` environment variables, or you can create z/OS jobs or started-task procedures for the Oracle GoldenGate processes. To use the environment variable `_BPX_JOBNAME`, at a minimum you should have read access to the RACF `FACILITY` class and `BPX.JOBNAME` name. For more details, see *Installing Oracle GoldenGate* and the IBM *z/OS System Services Planning* document.

5.5 Assigning WLM Velocity Goals

The user who starts the Manager process is typically the user by which other Oracle GoldenGate processes run. Oracle GoldenGate work appears as forked child processes of WLM subsystem type OMVS. Assign the Oracle GoldenGate processes their Workload Manager (WLM) velocity goals based on the following guidelines.

- Assign the Extract process that reads the transaction logs a medium velocity goal, one that is below the velocity of the main DB2 address spaces, but above the velocity of most online transactions, TSO/E sessions, and z/OS batch work. The higher the velocity goal, the more processor power that Extract will receive, and the less lag that it will experience.
- You can assign an initial-load Extract process a velocity goal, or you can treat it as a typical DB2 batch job. For more information about the initial-load processes, see *Administering Oracle GoldenGate*.
- You might need to assign the Replicat process a higher velocity goal. Although Replicat is a typical DB2 batch application, it might require more processing power to prevent backlogs and latency.
- You probably will run Oracle GoldenGate utilities, such as `DEFGEN` and LOGDUMP, only occasionally, so you can let them perform like the other UNIX terminal-oriented work.
- If executing stored procedures with the `SQLEXEC` command, make certain that they do not become a bottleneck for Oracle GoldenGate. Their priority should be close to that of the calling Extract or Replicat process. WLM executes them with that priority, but the z/OS system executes them under the priority of a stored procedure as defined by the DB2 and z/OS system programmers.
- If you run Oracle GoldenGate under the TSO/E `OMVS` command, the Oracle GoldenGate processes are subject to the system and WLM limits of the TSO/E user account, rather than those of the UNIX kernel. Very long TSO/E response times (up to 20 seconds), often with little service consumption, can be recorded for an OMVS user because of the way that OMVS polls for terminal input. This can affect those WLM goals that are based on response time.

You can use multiple WLM service classes for the Oracle GoldenGate processes. The following is an example of how to maintain relative priorities for Oracle GoldenGate and other work, from highest priority to the lowest:

1. z/OS system processes, including the UNIX kernel and IRLM.
2. DB2 for z/OS address spaces for the primary Extract group.
3. Primary Extract group configured for online or batch change synchronization, and any DB2 stored procedures that it calls.
4. z/OS transaction managers, such as CICS and IMS.
5. Collector (Server) for local Extract data pump, if used.
6. Local Extract data pump (reading from trail), if used.
7. Collector for remote trails (files received from a remote site). Such files include the QSAM file created with the Extract `RMTBATCH` parameter on a NonStop system.
8. Online Replicat groups and any DB2 stored procedures that they call.
9. Manager process (required only for startup of Oracle GoldenGate processes and trail cleanup).
10. GGSCI and other user UNIX and TSO/E terminal work.
11. Initial-load Extract and any DB2 stored procedures that it calls.
12. Initial-load Replicat and any DB2 stored procedures that it calls.
13. Other z/OS batch work.

5.6 Monitoring Processes

These sections provide information about monitoring Oracle GoldenGate with z/OS system facilities.

- Viewing Oracle GoldenGate Messages
- Identifying Oracle GoldenGate Processes
- Interpreting Statistics for Update Operations

5.6.1 Viewing Oracle GoldenGate Messages

If the system log process (`syslog` daemon `syslogd`) is running, USS routes Oracle GoldenGate messages to their configured destination by means of UNIX message priority. For more information about configuring `syslogd`, see the z/OS IP configuration documents and the *UNIX System Services Planning* document.

If `syslogd` is not running, Oracle GoldenGate writes its command output, status information, and error messages to the system console. You can redirect console messages to the Oracle GoldenGate USS session and to the Oracle GoldenGate report files by using the following UNIX command:

```
export _BPXK_JOBLOG=STDERR
```

5.6.2 Identifying Oracle GoldenGate Processes

The system management facility (SMF) typically creates a separate accounting record for each UNIX process, including Oracle GoldenGate processes. However, if a user invokes the UNIX shell by using the `OMVS` command with the default `SHAREAS` option, or if a user sets the environment variable `_BPX_SHAREAS` to `YES` , it could cause two or more processes to run in the same address space. SMF provides process identification only for the first process, but resource consumption is accumulated for all processes that are running. For Oracle GoldenGate, this means that the work probably will be recorded under the Manager process, which is named `mgr`.

If the DB2 accounting trace is also active to the SMF destination, DB2 will create an SMF accounting record for each of the following Oracle GoldenGate processes:

- Extract
- Replicat
- Manager, if performing maintenance on Oracle GoldenGate tables. Examples of Oracle GoldenGate tables are the marker table and the Replicat checkpoint table.
- GGSCI sessions that issue the Oracle GoldenGate `DBLOGIN` command to log into the database.

5.6.3 Interpreting Statistics for Update Operations

The actual number of DML operations that are executed on the DB2 database might not match the number of extracted DML operations that are reported by Oracle GoldenGate. DB2 does not log update statements if they do not physically change a row, so Oracle GoldenGate cannot detect them or include them in statistics.

5.7 Supporting Globalization Functions

Oracle GoldenGate provides globalization support and you should take into consideration when using this support.

- Replicating From a Source that Contains Both ASCII and EBCDIC
- Specifying Multi-Byte Characters in Object Names

5.7.1 Replicating From a Source that Contains Both ASCII and EBCDIC

When replicating to or from a DB2 source system to a target that has a different character set, some consideration must be given to the encoding of the character data on the DB2 source if it contains a mix of ASCII and EBCDIC data. Character set conversion by any given Replicat requires source data to be in a single character set.

The source character set is specified in the trail header. Thus, the Oracle GoldenGate trail can contain either ASCII or EBCDIC data, but not both. Unicode tables are processed without any special configuration and are exempt from the one-character set requirement.

With respect to a source that contains both character encoding types, you have the following options:

- You can use one Extract for all of your tables, and have it write the character data to the trail as either ASCII or as EBCDIC.
- You can use different Extracts: one Extract to write the ASCII character data to a trail, and another Extract to write the EBCDIC character data to a different trail. You then associate each trail with its own data pump process and Replicat process, so that the two data streams are processed separately.

To output the correct character set in either of those scenarios, use the `TRAILCHARSETASCII` and `TRAILCHARSETEBCDIC` parameters. The default is `TRAILCHARSETEBCDIC`. Without these parameters, ASCII and EBCDIC data are written to the trail as-is. When using these parameters, note the following:

- If used on a single-byte DB2 subsystem, these parameters cause Extract to convert all of the character data to either the ASCII or EBCDIC single-byte CCSID of the subsystem to which Extract is connected, depending on which parameter is used (except for Unicode, which is processed as-is).
- If used on a multi-byte DB2 subsystem, these parameters cause Extract to capture only ASCII or EBCDIC tables (and Unicode). Character data is written in either the ASCII or EBCDIC mixed CCSID (depending on the parameter used) of the DB2 z/OS subsystem to which Extract is connected.

5.7.2 Specifying Multi-Byte Characters in Object Names

If the name of a schema, table, column, or stored procedure in a parameter file contains a multi-byte character, the name must be double-quoted. For more information about specifying object names, see *Administering Oracle GoldenGate*.

6

Preparing the DB2 for z/OS Transaction Logs for Oracle GoldenGate

Learn how to configure the DB2 transaction logging to support data capture by Oracle GoldenGate Extract.

Topics:

6.1 Making Transaction Data Available

Oracle GoldenGate can extract DB2 transaction data from the active and archived logs. Follow these guidelines to configure the logs so that Extract can capture data.

6.1.1 Enabling Change Capture

Follow these steps to configure DB2 to log data changes in the expanded format that is supplied by the `DATA CAPTURE CHANGES` feature of the `CREATE TABLE` and `ALTER TABLE` commands. This format provides Oracle GoldenGate with the entire before and after images of rows that are changed with update statements.

1. From the Oracle GoldenGate directory, run GGSCI.
2. Log on to DB2 from GGSCI as a user that has `ALTER TABLE` privileges.

   ```
   DBLOGIN SOURCEDB DSN, USERID user[, PASSWORD password][, encryption_options]
   ```
3. Issue the following command. where *`table`* is the fully qualified name of the table. You can use a wildcard to specify multiple table names but not owner names.

   ```
   ADD TRANDATA table
   ```

   By default, `ADD TRANDATA` issues the following command:

   ```
   ALTER TABLE name DATA CAPTURE CHANGES;
   ```

6.1.2 Enabling Access to Log Records

Activate DB2 Monitor Trace Class 1 ("`TRACE(MONITOR) CLASS(1)` ") so that DB2 allows Extract to read the active log. The default destination of `OPX` is sufficient, because Oracle GoldenGate does not use a destination.

To Start the Trace Manually

1. Log on to DB2 as a DB2 user who has the `TRACE` privilege or at least `SYSOPR` authority.
2. Issue the following command:

   ```
   start trace(monitor) class(1) scope(group)
   ```

To Start the Trace Automatically When DB2 is Started

Do either of the following:

- Set `MONITOR TRACE` to "`YES`" on the `DSNTIPN` installation tracing panel.
- Set '`DSN6SYSP MON=YES`' in the `DSNTIJUZ` installation job, as described in the *DB2 UDB Installation Guide*.

> **Note:**
>
> The primary authorization ID, or one of the secondary authorization IDs, of the ODBC plan executor also must have the `MONITOR2` privilege.

6.1.3 Sizing and Retaining the Logs

When tables are defined with `DATA CAPTURE CHANGES`, more data is logged than when they are defined with `DATA CAPTURE NONE`. If any of the following is true, you might need to increase the number and size of the active and archived logs.

- Your applications generate large amounts of DB2 data.
- Your applications have infrequent commits.
- You expect to stop Extract for long periods of time.
- Your network is unreliable or slow.

To control log retention, use the `DSN6LOGP MAXARCH` system parameter in the `DSNTIJUZ` installation job.

Retain enough log data so that Extract can start again from its checkpoints after you stop it or after an unplanned outage. Extract must have access to the log that contains the start of the oldest uncommitted unit of work, and all logs thereafter.

If data that Extract needs during processing was not retained, either in online or archived logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which log data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

Note:

The IBM documentation makes recommendations for improving the performance of log reads. In particular, you can use large log output buffers, large active logs, and make archives to disk.

6.1.4 Using Archive Logs on Tape

Oracle GoldenGate can read DB2 archive logs on tape, but it will degrade performance. For example, DB2 reserves taped archives for a single recovery task. Therefore, Extract would not be able to read an archive tape that is being used to recover a table until the recovery is finished. You could use DFHSM or an equivalent tools to move the archive logs in a seamless manner between online DASD storage and tape, but Extract will have to wait until the transfer is finished. Delays in Extract processing increase the latency between source and target data.

6.1.5 Controlling Log Flushes

When reading the transaction log, Extract does not process a transaction until it captures the commit record. If the commit record is on a data block that is not full, it cannot be captured until more log activity is generated to complete the block. The API that is used by Extract to read the logs only retrieves full physical data blocks.

A delay in receiving blocks that contain commits can cause latency between the source and target data. If the applications are not generating enough log records to fill a block, Extract generates its own log records by issuing `SAVEPOINT` and `COMMIT` statements, until the block fills up one way or the other and is released.

In a data sharing group, each API call causes DB2 to flush the data blocks of all active members, eliminating the need for Extract to perform flushes.

To prevent Extract from performing flushes, use the Extract parameter `TRANLOGOPTIONS` with the `NOFLUSH` option.

Part IV
Using Oracle GoldenGate with MySQL

Oracle GoldenGate for MySQL supports replication from a MySQL source database to a MySQL target database or to a supported database of another type to perform an initial load or change data replication.

This part describes tasks for configuring and running Oracle GoldenGate on a MySQL database.

Topics:

7

Understanding What's Supported for MySQL

This chapter contains information on database and table features supported by Oracle GoldenGate.

Topics:

7.1 Character Sets in MySQL

MySQL provides a facility that allows users to specify different character sets at different levels.

Level	Example
Database	`create database test charset utf8;`
Table	`create table test( id int, name char(100)) charset utf8;`
Column	`create table test ( id int, name1 char(100) charset gbk, name2 char(100) charset utf8));`

Limitations of Support

- When you specify the character set of your database as utf8mb4/utf8, the default collation is `utf8mb4_unicode_ci`/`utf8_general_ci`. If you specify `collation_server=utf8mb4_bin`, the database interprets the data as binary. For example, specifying the `CHAR` column length as four means that the byte length returned is 16 (for utf8mb4) though when you try to insert data more than four bytes the target database warns that the data is too long. This is the limitation of database so Oracle GoldenGate does not support binary collation. To overcome this issue, specify `collation_server=utf8mb4_bin` when the character set is utf8mb4 and `collation_server=utf8_bin` for utf8.
- The following character sets are not supported:

  ```
  armscii8
  keybcs2
  utf16le
  geostd8
  ```

7.2 Supported MySQL Data Types

MySQL supports the following data types:

- `CHAR`
- `VARCHAR`
- `INT`
- `TINYINT`
- `SMALL INT`
- `MEDIUM INT`
- `BIG INT`
- `DECIMAL`
- `FLOAT`
- `DOUBLE`
- `DATE`
- `TIME`
- `YEAR`
- `DATETIME`
- `TIMESTAMP`
- `BINARY`
- `VARBINARY`
- `TEXT`
- `TINYTEXT`
- `MEDIUMTEXT`
- `LONGTEXT`
- `BLOB`
- `TINYBLOB`
- `MEDIUMBLOB`
- `LONGBLOB`
- `ENUM`
- `BIT(M)`
- Limitations and Clarifications

7.2.1 Limitations and Clarifications

When running Oracle GoldenGate for MySQL, be aware of the following:

- Oracle GoldenGate does not support `BLOB` or `TEXT` types when used as a primary key.

- Oracle GoldenGate supports UTF8 and UCS2 character sets. UTF8 data is converted to UTF16 by Oracle GoldenGate before writing it to the trail.
- UTF32 is not supported by Oracle GoldenGate.
- Oracle GoldenGate supports a `TIME` type range from 00:00:00 to 23:59:59.
- Oracle GoldenGate supports timestamp data from `0001/01/03:00:00:00` to `9999/12/31:23:59:59`. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the time zone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support negative dates.
- The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.
- When you use `ENUM` type in non-strict `sql_mode`, the non-strict `sql_mode` does not prevent you from entering an invalid `ENUM` value and an error will be returned. To avoid this situation, do one of the following:
  - Use `sql_mode` as `STRICT` and restart Extract. This prevents users from entering invalid values for any of the data types. An IE user can only enter valid values for those data types.
  - Continue using non-strict `sql_mode`, but do not use `ENUM` data types.
  - Continue using non-strict `sql_mode` and use `ENUM` data types with valid values in the database. If you specify invalid values, the database will silently accept them and Extract will abend.
- To preserve transaction boundaries for a MySQL target, create or alter the target tables to the InnoDB transactional database engine instead of the MyISAM engine. MyISAM will cause Replicat records to be applied as they are received, which does not guarantee transaction integrity even with auto-commit turned off. You cannot roll back a transaction with MyISAM.
- Extraction and replication from and to views is not supported.
- Transactions applied by the slave are logged into the relay logs and not into the slave's `binlog`. If you want a slave to write transactions the `binlog` that it receives from the master , you need to start the replication slave with the log slave-updates option as 1 in `my.cnf`. This is in addition to the other binary logging parameters. After the master's transactions are in the slave's `binlog`, you can then setup a regular capture on the slave to capture and process the slave's `binlog`.

7.3 Supported Objects and Operations for MySQL

Oracle GoldenGate for MySQL supports the following objects and operations:

- Basic extraction and replication of DDL (data definition language) operations for MySQL 5.7.10 and later. Only the `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` operations are supported.
- Oracle GoldenGate supports the extraction and replication of transactional tables.

- Oracle GoldenGate supports transactional tables up to the full row size and maximum number of columns that are supported by MySQL and the database storage engine that is being used. InnoDB supports up to 1017 columns.
- Oracle GoldenGate supports the `AUTO_INCREMENT` column attribute. The increment value is captured from the binary log by Extract and applied to the target table in a Replicat insert operation.
- Oracle GoldenGate supports the following DML operations on source and target database transactional tables:
  - Insert operation
  - Update operation (compressed included)
  - Delete operation (compressed included); cascade delete queries result in the deletion of the child of the parent operation
  - Truncate operation
- Oracle GoldenGate can operate concurrently with MySQL native replication.
- Oracle GoldenGate supports the `DYNSQL` feature for MySQL.

Note:

XA transactions are not supported for capture and any XA transactions logged in `binlog` cause Extract to abend. So, you must not use XA transactions against a database that Extract is configured to capture. If XA transactions are being used for databases that are not configured for Oracle GoldenGate capture, then exclude those databases from logging into MySQL binary logs by using the parameter `binlog-ignore-db` in the MySQL server configuration file.

Limitations on Automatic Heartbeat Table support are as follows:

- Ensure that the database in which the heartbeat table is to be created already exists to avoid errors when adding the heartbeat table.
- In the heartbeat history lag view, the information in fields like `heartbeat_received_ts`, `incoming_heartbeat_age`, and `outgoing_heartbeat_age` are shown with respect to the system time. You should ensure that the operating system time is setup with the correct and current time zone information.

7.4 Non-Supported MySQL Data Types

Oracle GoldenGate for MySQL does not support the following data types:

`XML`, `SET`, all spatial types (Geometry and so on), JSON, Interval.

Note:

Extract abends if it is configured to capture from tables that contain any of the unsupported data types, so ensure that Extract is not configured to capture from tables containing columns of unsupported data types.

8

Preparing and Configuring the System for Oracle GoldenGate

Learn about how to prepare the system for running Oracle GoldenGate and how to configure it with your MySQL database.
Topics:

8.1 Ensuring Data Availability

Retain enough binary log data so that if you stop Extract or there is an unplanned outage, Extract can start again from its checkpoints. Extract must have access to the binary log that contains the start of the oldest uncommitted unit of work, and all binary logs thereafter. The recommended retention period is at least 24 hours worth of transaction data, including both active and archived information. You might need to do some testing to determine the best retention time given your data volume and business requirements.

If data that Extract needs during processing was not retained, either in active or backup logs, one of the following corrective actions might be required:

- Alter Extract to capture from a later point in time for which binary log data is available (and accept possible data loss on the target).
- Resynchronize the source and target tables, and then start the Oracle GoldenGate environment over again.

To determine where the Extract checkpoints are, use the `INFO EXTRACT` command. For more information, see INFO EXTRACT in *Reference for Oracle GoldenGate*

8.2 Setting Logging Parameters

To capture from the MySQL transaction logs, the Oracle GoldenGate Extract process must be able to find the index file. index file in turn contains the paths of all binary log files.

> **Note:**
>
> Extract expects that all of the table columns are in the binary log. As a result, only `binlog_row_image` set as `full` is supported and this is the default. Other values of `binlog_row_image` are not supported.

In MySQL 5.7, the `server_id` option must be specified along with `log-bin`, otherwise the server will not start. For MySQL 8.0, the `server_id` is enabled by default.

Extract checks the following parameter settings to get this index file path:

1. Extract `TRANLOGOPTIONS` parameter with the `ALTLOGDEST` option: If this parameter specifies a location for the log index file, Extract accepts this location over any default that is specified in the MySQL Server configuration file. When `ALTLOGDEST` is used, the binary log index file must also be stored in the specified directory. This parameter should be used if the MySQL configuration file does not specify the full index file path name, specifies an incorrect location, or if there are multiple installations of MySQL on the same machine

   To specify the index file path with `TRANLOGOPTIONS` with `ALTLOGDEST`, use the following command format on Windows:

   ```
   TRANLOGOPTIONS ALTLOGDEST "C:\Program Files\MySQL\logs\binlog.index"
   ```

   On Linux, use this format:

   ```
   TRANLOGOPTIONS ALTLOGDEST "/mnt/rdbms/mysql/data/logs/binlog.index"
   ```

   To capture from a remote server or in case of remote capture, you only need to specify the `REMOTE` option instead of the index file path on the remote server. For remote capture on both Windows and Linux, specify the following in the Extract parameter file:

   ```
   TRANLOGOPTIONS ALTLOGDEST REMOTE
   ```

2. The MySQL Server configuration file: The configuration file stores default startup options for the MySQL server and clients. On Windows, the name of the configuration file is `my.ini`. On other platforms, it is `my.cnf`. In the absence of `TRANLOGOPTIONS` with `ALTLOGDEST`, Extract gets information about the location of the log files from the configuration file. However, even with `ALTLOGDEST`, these Extract parameters must be set correctly:

   - `binlog-ignore-db=oggddl`: This prevents DDL logging history table entries in the `binlog` and is set in the `my.cnf` or `my.ini` file.
   - `log-bin`: This parameter is used to enable binary logging. This parameter also specifies the location of the binary log index file and is a required parameter

for Oracle GoldenGate, even if `ALTLOGDEST` is used. If log-bin is not specified, binary logging will be disabled and Extract returns an error.

- `log-bin-index`: This parameter specifies the location of the binary log index. If it is not used, Extract assumes that the index file is in the same location as the log files. If this parameter is used and specifies a different directory from the one that contains the binary logs, the binary logs must not be moved once Extract is started.
- `max_binlog_size`: This parameter specifies the size, in bytes, of the binary log file.

> **Note:**
>
> The server creates a new binary log file automatically when the size of the current log reaches the `max_binlog_size` value, unless it must finish recording a transaction before rolling over to a new file.

- `binlog_format`: This parameter sets the format of the logs. It must be set to the value of `ROW`, which directs the database to log DML statements in binary format. Any other log format (`MIXED` or `STATEMENT`) causes Extract to abend.

> **Note:**
>
> MySQL binary logging does not allow logging to be enabled or disabled for specific tables. It applies globally to all tables in the database.

To locate the configuration file, Extract checks the `MYSQL_HOME` environment variable: If `MYSQL_HOME` is set, Extract uses the configuration file in the specified directory. If `MYSQL_HOME` is not set, Extract queries the `information_schema.global_variables` table to determine the MySQL installation directory. If a configuration file exists in that directory, Extract uses it.

3. For MariaDB version 10.2 and later, Oracle GoldenGate works in the same way as for MySQL but a new variable needs to be configured in the `my.cnf` or `my.ini` file. The variable that needs to be added is `"binlog-annotate-row-events=OFF"`. Restart MariaDB after configuring this variable and then start the Extract process.

8.3 Adding Host Names

Oracle GoldenGate gets the name of the database it is supposed to connect to from the `SOURCEDB` parameter. A successful connection depends on the localhost entry being properly configured in the system host file. To avoid issues that arise from improper local host configuration, you can use `SOURCEDB` in the following format:

```
SOURCEDB dbname@hostname:port, USERID mysqluser, PASSWORD welcome
```

The `dbname` is the name of the MySQL instance,`hostname` is the name or IP address, `port` is the port number of the MySQL instance. If using an unqualified host name, that name must be properly configured in the DNS database. Otherwise, use the fully qualified host name, for example `myhost.company.com`.

8.4 Setting the Session Character Set

The `GGSCI`, Extract and Replicat processes use a session character set when connecting to the database. For MySQL, the session character set is taken from the `SESSIONCHARSET` option of `SOURCEDB` and `TARGETDB`. Make certain you specify a session character set in one of these ways when you configure Oracle GoldenGate.

8.5 Preparing Tables for Processing

This section describes how to prepare the tables for processing. Table preparation requires these tasks:

- Assigning Row Identifiers
- Limiting Row Changes in Tables That Do Not Have a Key
- Disabling Triggers and Cascade Constraints

8.5.1 Assigning Row Identifiers

Oracle GoldenGate requires some form of unique row identifier on the source and target tables to locate the correct target rows for replicated updates and deletes.

- How Oracle GoldenGate Determines the Kind of Row Identifier to Use
- Tables with a Primary Key Derived from a Unique Index
- How to Specify Your Own Key for Oracle GoldenGate to Use

8.5.1.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If none of the preceding key types exist (even though there might be other types of keys defined on the table) Oracle GoldenGate constructs a pseudo key of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration.

Note:

If there are other, non-usable keys on a table or if there are no keys at all on the table, Oracle GoldenGate logs an appropriate message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient `WHERE` clause.

8.5.1.2 Tables with a Primary Key Derived from a Unique Index

In the absence of a primary key on a table, MySQL will promote a unique index to primary key if the indexed column is `NOT NULL`. If there are more than one of these not-null indexes, the first one that was created becomes the primary key. To avoid Replicat errors, create these indexes in the same order on the source and target tables.

For example, assume that source and target tables named `ggvam.emp` each have columns named first, middle, and last, and all are defined as `NOT NULL`. If you create unique indexes in the following order, Oracle GoldenGate will abend on the target because the table definitions do not match.

Source:

```
mysql> create unique index uq1 on ggvam.emp(first);
mysql> create unique index uq2 on ggvam.emp(middle);
mysql> create unique index uq3 on ggvam.emp(last);
```

Target:

```
mysql> create unique index uq1 on ggvam.emp(last);
mysql> create unique index uq2 on ggvam.emp(first);
mysql> create unique index uq3 on ggvam.emp(middle);
```

The result of this sequence is that MySQL promotes the index on the source "first" column to primary key, and it promotes the index on the target "last" column to primary key. Oracle GoldenGate will select the primary keys as identifiers when it builds its metadata record, and the metadata will not match. To avoid this error, decide which column you want to promote to primary key, and create that index first on the source and target.

8.5.1.3 How to Specify Your Own Key for Oracle GoldenGate to Use

If a table does not have one of the preceding types of row identifiers, or if you prefer those identifiers not to be used, you can define a substitute key if the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key will override any existing primary or unique key that Oracle GoldenGate finds.

8.5.2 Limiting Row Changes in Tables That Do Not Have a Key

If a target table does not have a primary key or a unique key, duplicate rows can exist. In this case, Oracle GoldenGate could update or delete too many target rows, causing the source and target data to go out of synchronization without error messages to alert you. To limit the number of rows that are updated, use the `DBOPTIONS` parameter with the `LIMITROWS` option in the Replicat parameter file. `LIMITROWS` can increase the performance of Oracle GoldenGate on the target system because only one row is processed.

8.5.3 Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on the target tables, or alter them to ignore changes made by the Oracle GoldenGate database user. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

8.6 Changing the Log-Bin Location

Modifying the binary log location by using the `log-bin` variable in the MySQL configuration file might result in two different path entries inside the index file, which could result in errors. To avoid any potential errors, change the log-bin location by doing the following:

1. Stop any new DML operations.
2. Let the extract finish processing all of the existing binary logs. You can verify this by noting when the checkpoint position reaches the offset of the last log.
3. After Extract finishes processing the data, stop the Extract group and, if necessary, back up the binary logs.
4. Stop the MySQL database.
5. Modify the `log-bin` path for the new location.
6. Start the MySQL database.
7. To clean the old log name entries from index file, use `flush master` or `reset master` (based on your MySQL version).
8. Start Extract.

8.7 Configuring Bi-Directional Replication

In a bi-directional configuration, there are Extract and Replicat processes on both the source and target systems to support the replication of transactional changes on each system to the other system. To support this configuration, each Extract must be able to filter the transactions applied by the local Replicat, so that they are not recaptured and sent back to their source in a continuous loop. Additionally, `AUTO_INCREMENT` columns must be set so that there is no conflict between the values on each system.

1. Configure Oracle GoldenGate for high availability or active-active replication according to the instructions in the Overview of Replicat in *Administering Oracle GoldenGate*

2. To filter out Replicat operations in a bi-directional configuration so that the applied operations are not captured and looped back to the source again, take the following steps on each MySQL database:
   - Configure each Replicat process to use a checkpoint table. Replicat writes a checkpoint to this table at the end of each transaction. You can use one global checkpoint table or one per Replicat process See Overview of Replicat in *Administering Oracle GoldenGate*.
   - Specify the name of the checkpoint table with the `FILTERTABLE` option of the `TRANLOGOPTIONS` parameter in the Extract parameter file. The Extract process will ignore transactions that end with an operation to the specified table, which should only be those of Replicat.

     > **Note:**
     >
     > Although optional for other supported databases as a means of enhancing recovery, the use of a checkpoint table is required for MySQL when using bi-directional replication (and likewise, will enhance recovery).

3. Edit the MySQL server configuration file to set the `auto_increment_increment` and `auto_increment_offset` parameters to avoid discrepancies that could be caused by the bi-directional operations. The following illustrates these parameters, assuming two servers: **ServerA** and **ServerB**.

   ServerA:

   ```
   auto-increment-increment = 2
   auto-increment-offset = 1
   ```

   ServerB:

   ```
   auto-increment-increment = 2
   auto-increment-offset = 2
   ```

8.8 Oracle GoldenGate for MySQL: Remote Capture

Oracle GoldenGate's remote capture for MySQL is used to capture transaction log data from a MySQL database located remotely to the Oracle GoldenGate installation.

Database Server Configuration

For remote capture to work, configure the MySQL server as follows:

1. Grant access permissions to the Oracle GoldenGate remote capture user.

   Run the following MySQL commands to create and grant permissions to the remote user on MySQL Server.

   ```
   mysql > CREATE USER 'username'@'host' IDENTIFIED BY '<Password>';
   mysql > GRANT ALL PRIVILEGES ON *.* TO 'username'@'host’ WITH GRANT
   OPTION;
   mysql > FLUSH PRIVILEGES;
   ```

2. The `server_id` value of the remote MySQL Server should be greater than 0. This value can be verified by issuing the following command on the MySQL remote server:

   ```
   mysql > show variables like ‘server_id’;
   ```

   If the `server_id` value is 0, modify the `my.cnf` configuration file to set to a value greater than 0.

Oracle GoldenGate Configuration

Oracle GoldenGate configuration has the following steps:

1. Provide remote MySQL server’s connection details in the connection parameters of capture `.prm` file.

   ```
   SOURCEDB remotedb@mysqlserver.company.com, USERID remote, PASSWORD
   welcome
   ```

2. In the capture parameter file, specify the following:

   ```
   TRANLOGOPTIONS ALTLOGDEST REMOTE
   ```

Limitations of Oracle GoldenGate Remote Capture for MySQL

Co-existence of Oracle GoldenGate for MySQL remote capture with the MySQL’s native replication slave is supported with following conditions and limitations:

- The native replication slave should be assigned a different server_id than the currently running slaves. The slave server_id values can be seen using the following MySQL command on the master server.

  ```
  mysql> show slave hosts;
  ```

  – If the Oracle GoldenGate capture abends with error "`A slave with the same server_uuid or server_id as this slave has connected to the master`", then change the capture's name and restart the capture.
  – If the native replication slave dies with the error "`A slave with the same server_uuid or server_id as this slave has connected to the master`", then change the native replication slave’s `server_id` and restart it.
- DDL replication is not supported for the remote capture.
- Remote capture is supported only on the Linux 64-bit platform and not on Windows. But OGG remote capture on Linux can capture from the MySQL server running on remote Windows machine.

8.9 Capturing using a MySQL Replication Slave

You can configure a MySQL replication slave to capture the master's binary log events from the slave.

Typically, the transactions applied by the slave are logged into the relay logs and not into the slave's `binlog`. For the slave to write transactions in its `binlog`, that it receives from the master , you must start the replication slave with the `log-slave-updates` option as `1` in `my.cnf` in conjunction with the other binary logging parameters for Oracle GoldenGate. After the master's transactions are in the slave's `binlog` , you can set up a regular Oracle GoldenGate capture on the slave to capture and process the slave's `binlog`.

8.10 Other Oracle GoldenGate Parameters for MySQL

The following parameters may be of use in MySQL installations, and might be required if non-default settings are used for the MySQL database. Other Oracle GoldenGate parameters will be required in addition to these, depending on your intended business use and configuration.

Parameter	Description
`DBOPTIONS` with `CONNECTIONPORT` *`port_number`*	Required to specify to the VAM the TCP/IP connection port number of the MySQL instance to which an Oracle GoldenGate process must connect if MySQL is not running on the default of 3306. `DBOPTIONS CONNECTIONPORT 3307`
`DBOPTIONS` with `HOST` *`host_id`*	Specifies the DNS name or IP address of the system hosting MySQL to which Replicat must connect.
`DBOPTIONS` with `ALLOWLOBDATATRUNCATE`	Prevents Replicat from abending when replicated LOB data is too large for a target MySQL `CHAR`, `VARCHAR`, `BINARY` or `VARBINARY` column.

`SOURCEDB with USERID` and `PASSWORD`	Specifies database connection information consisting of the database, user name and password to use by an Oracle GoldenGate process that connects to a MySQL database. If MySQL is not running on the default port of 3306, you must specify a complete connection string that includes the port number: `SOURCEDB` *`dbname@hostname:port`*, `USERID` *`user`*, `PASSWORD` *`password`*.Example:
	`SOURCEDB mydb@mymachine:3307, USERID myuser, PASSWORD mypassword`
	If you are not running the MySQL database on port 3306, you must also specify the connection port of the MySQL database in the `DBLOGIN` command when issuing commands that affect the database through GGSCI:
	`DBLOGIN SOURCEDB` *`dbname@hostname`*`:`*`port`*`, USERID` *`user`*`, PASSWORD` *`password`*
	For example:
	`GGSCI> DBLOGIN SOURCEDB mydb@mymachine:3307, USERID myuser, PASSWORD mypassword`

Parameter	Description
`SQLEXEC`	To enable Replicat to bypass the MySQL connection timeout, configure the following command in a `SQLEXEC` statement in the Replicat parameter file. `SQLEXEC "select CURRENT_TIME();" EVERY` *`n`* `MINUTES` **Where**: *n* is the maximum interval after which you want Replicat to reconnect. The recommended connection timeout 31536000 seconds (365 days).

8.11 Positioning Extract to a Specific Start Point

You can position the `ADD EXTRACT` and `ALTER EXTRACT` commands to a specific start point in the transaction logs with the following command.

```
{ADD | ALTER EXTRACT} group, VAM, LOGNUM log_num, LOGPOS log_pos
```

- *`group`* is the name of the Oracle GoldenGate Extract group for which the start position is required.
- *`log_num`* is the log file number. For example, if the required log file name is `test.000034`, this value is 34. Extract will search for this log file.
- *`log_pos`* is an event offset value within the log file that identifies a specific transaction record. Event offset values are stored in the header section of a log record. To position at the beginning of a `binlog` file, set the `log_pos` as 4. The `log_pos` 0 or 1 are not valid offsets to start reading and processing.

In MySQL logs, an event offset value can be unique only within a given binary file. The combination of the position value and a log number will uniquely identify a transaction record and cannot exceed a length of 37. Transactional records available after this position within the specified log will be captured by Extract. In addition, you can position an Extract using a timestamp.

9 Using DDL Replication

Learn how to install, use, configure, and remove DDL replication.
Data Definition Language (DDL) statements (operations) are used to define MySQL database structures or schema. You can use these DDL statements for data replication between MySQL source and target databases. MySQL DDL specifics are found in the MySQL documentation at https://dev.mysql.com/doc/.

Topics:

9.1 DDL Configuration Prerequisites and Considerations

The prerequisites for configuring DDL replication are as follows:

- DDL replication is supported for MySQL 5.7.10 and greater.
- Bidirectional filtering for DDL replication is not supported.
- Remote capture implementation doesn’t support DDL replication.
- Oracle GoldenGate DDL replication uses two plug-ins as a shared library, `ddl_rewriter` and `ddl_metadata`, which must be installed on your MySQL server before Oracle GoldenGate replication starts.
- The standalone application, Oracle GoldenGate `metadata_server`, must be running to capture the DDL metadata.
- The `history` table under the new `oggddl` database (`oggddl.history`). This metadata history table is used to store and retrieve the DDL metadata history. The history table records must be ignored from being logged into the binary log so you must specify `binlog-ignore-db=oggddl` in the `my.cnf` file.
- You should not manually drop the `oggddl` database or the `history` table because all DDL statements that run after this event will be lost.
- You should not stop the `metadata_server` during DDL capture as all the DDL statements that run after this event will be lost.
- You should not manually remove the `ddl_rewriter` and the `ddl_metadata` plugins during DDL capture because all DDL statements that run after this event will be lost.

- DDL executed within the stored procedure is *not* supported. For example , a DDL executed as in the following is *not* supported.

  ```
  CREATE PROCEDURE atssrc.generate_data()
  BEGIN
  DECLARE i INT DEFAULT 0;
  WHILE i < 800 DO
  SET i = i + 1;
  IF (i = 100) then
  alter table atssrc.`ddl6` add col2 DATE after id;
  ELSEIF (i = 200) then
  alter table atssrc.`ddl6` add col3 DATETIME after datetime;
  ELSEIF (i = 300) then
  alter table atssrc.`ddl6` add `col4` timestamp NULL DEFAULT NULL after
  channel;
  ELSEIF (i = 400) then
  alter table atssrc.`ddl6` add col5 YEAR after value;
  END IF;
  END WHILE;
  END$$
  DELIMITER ;
  call atssrc.generate_data();
  ```

- By design, the heartbeat DDLs are ignored by the capture and you should create the heartbeat tables manually at the target.

9.2 Installing DDL Replication

To install DDL replication, you run the installation script that is provided with Oracle GoldenGate as the replication user. This user must have `Create`, `Insert`,`Select`, `Delete`, `Drop`, and `Truncate` database privileges. Additionally, this user must have write permission to copy the Oracle GoldenGate plugin in the MySQL plugin directory. For example, the MySQL plugin are typically in `/usr/lib64/mysql/plugin/`.

The installation script options are `install`, `uninstall`, `start`, `stop`, and `restart`.

The command to install DDL replication uses the install option, user id, password, and port number respectively:

```
bash-3.2$ ./ddl_install.sh install-option user-id password port-number
```

For example:

```
bash-3.2$ ./ddl_install.sh install root welcome 3306
```

The DDL replication installation script completes the following tasks:

1. Ensures that you have a supported MySQL server version installed. DDL replication is supported for MySQL 5.7.10 and greater.
2. Locates the MySQL plugin directory.
3. Ensures that the `ddl_rewriter`, `ddl_metadata` plugins and the `metadata_server` files exist. If these files are not found, then an error message appears and the installation exits.

4. Ensures that the plugins are already installed. If installed, the script exits with a message requesting you to uninstall first and then reinstall.
5. Stops the `metadata_server` if it is running.
6. Deletes the `oggddl.history` table if it exists.
7. Starts the `metadata_server` as a daemon process.
8. Installs the `ddl_rewriter` and `ddl_metadata` plugins.

9.3 Using the Metadata Server

You can use the following options with the metadata server:

- You must have the Oracle GoldenGate `metadata_server` running to capture the DDL metadata.
- Run the install script with `start` option to start the metadata server.
- Run the install script with `stop` option to stop the metadata server.
- Run the install script with `restart` option to stop the running metadata server and start again.
- Oracle GoldenGate DDL replication uses two plugins as a shared library, `ddl_rewriter` and `ddl_metadata`, both of which must be installed on your MySQL server before Oracle GoldenGate replication starts.
- The `oggddl.history` metadata history table is used to store and retrieve the DDL metadata history.

There is a single history table and metadata server for each MySQL server. If you want to issue and capture DDLs from multiple instances of an Extract process on the same database server at the same time, there is a possibility of conflict between accessing and populating the metadata history table. Oracle recommends that you do not run and capture DDLs using multiple Extract instances on the same MySQL server.

9.4 Using DDL Filtering for Replication

The following options are supported for MySQL DDL replication:

Option	Description
`DDL INCLUDE OPTYPE CREATE OBJTYPE TABLE;`	Include create table.
`DDL INCLUDE OBJNAME ggvam.*`	Include tables under the `ggvam`database.
`DDL EXCLUDE OBJNAME ggvam.emp*;`	Exclude all the tables under the `ggvam` database and table name starting with the `emp`wildcard.
`DDL INCLUDE INSTR ‘XYZ’`	Include DDL that contains this string.
`DDL EXCLUDE INSTR ‘WHY’`	Excludes DDL that contains this string.

Option	Description
`DDL INCLUDE MAPPED`	MySQL DDL uses this option and should be used as the default for Oracle GoldenGate MySQL DDL replication. `DDL INCLUDE ALL` and `DDL` are not supported.
`DDL EXCLUDE ALL`	Default option.

For a full list of options, see DDL in *Reference for Oracle GoldenGate*.

Using DDL Statements and Options

- `INCLUDE` (default) means include all objects that fit the rest of the description. `EXCLUDE` means to omit items that fit the description. Exclude rules take precedence over include rules.
- `OPTYPE` specifies the types of operations to be included or excluded. You can use `CREATE` and `ALTER`. Multiple `OPTYPE` can be specified using parentheses. For example, `optype (create, alter)`. The asterisk (*) wildcard can be specified to indicate all operation types, and this is the default.
- `OBJTYPE` specifies the `TABLE` operations to include or exclude. The wildcard can be specified to indicate all object types, and this is the default.
- `OBJNAME` specifies the actual object names to include or exclude. For example, `eric.*`. Wildcards are specified as in other cases where multiple tables are specified. The default is `*`.
- `String` indicates that the rule is true if any of the strings in `stringspec` are present (or false if `excludestring` is specified and the `stringspec` is present). If multiple `string` entries are made, at least one entry in each `stringspec` must be present to make the rule evaluate true.

  For example:

  ```
  ddlops string (“a”, “b”), string (“c”) evaluates true if string “a” OR
  “b” is present, AND string “c” is present
  ```

- `local` is specified if you want the rule to apply only to the current Extract trail (the Extract trail to which the rule applies must precede this `ddlops` specification).
- The semicolon is required to terminate the parameter entry.

  For example:

  ```
  ddl optype (create, drop), objname (eric.*);
  ddl exclude objname (eric.tab*);
  exttrail a;
  exttrail b;
  ddl optype (create), objname (joe.*), string (“abc”, “xyz”) local;
  ddl optype (alter), objtype (index);
  ```

  In this preceding example, the `exttrail a` gets creates and drops for all objects that belong to `eric`, except for objects that start with `tab`, `exttrail a` also gets all alter index statements, unless the index name begins with `tab` (the rule is global even though it’s included in `exttrail b`). `exttrail b` gets the same objects as `a`, and it also gets all creates for objects that belong to `joe` when the string `abcor` `xyz` is present in the DDL text. The `ddlops.c` module stores all DDL operation parameters and executes related rules.

Additionally, you can use the `DDLOPTIONS` parameter to configure aspects of DDL processing other than filtering and string substitution. You can use multiple `DDLOPTIONS` statements and Oracle recommends using one. If you are using multiple `DDLOPTIONS` statements, then make each of them unique so that one does not override the other. Multiple `DDLOPTIONS` statements are executed in the order listed in the parameter file.

See DDL and DDLOPTIONS.

9.5 Troubleshooting DDL Replication

DDL replication relies on a metadata history table and the metadata plugin and server. To troubleshoot when DDL replication is enabled, the history table contents and the metadata plugin server logs are required.

You can use the `mysqldump` command to generate the history table dump using one of the following examples:

```
mysqldump [options] database [tables]
mysqldump [options] --databases [options] DB1 [DB2 DB3...]
mysqldump [options] --all-databases [options]
```

For example, `bash-3.2$ mysqldump -uroot -pwelcome oggddl history > outfile`

The metadata plugins and server logs are located in the MySQL and Oracle GoldenGate installation directories respectively.

If you find an error in the log files, you need to ensure that the metadata server is running.

9.6 Uninstalling DDL Replication

If you no longer want to capture the DDL events, then you can use the same install script and select the `uninstall` option to disable the DDL setup. Also, any Extract with DDL parameters should be removed or disabled. If you want to capture the DDL again, you can run the install script again. You should take care when multiple instances of the capture process is running on the same instance of your MySQL server. The DDL setup should *not* be disturbed or uninstalled when multiple capture processes are running and when at most one capture is designed to capture the DDL statement.

Use the installation script with the `uninstall` option to uninstall DDL Replication. For example:

```
bash-3.2$ ./ddl_install.sh uninstall root welcome 3306
```

The script performs the following tasks:

1. Uninstalls the `ddl_rewriter` and `ddl_metadata` plugins.
2. Deletes the `oggddl.history` table if exists.
3. Removes the plugins from MySQL plugin directory.

4. Stops the `metadata_server` if it is running.

Part V

Using Oracle GoldenGate for Teradata

With Oracle GoldenGate for Teradata, you can deliver initial load and transactional data from other supported Oracle GoldenGate sources, such as an Oracle database.

Oracle GoldenGate for Teradata supports data filtering, mapping, and transformations unless noted otherwise in this documentation.

This part describes tasks for configuring and running Oracle GoldenGate for Teradata.

Topics

Overview of Oracle GoldenGate for Teradata

Oracle GoldenGate for Teradata supports the filtering, mapping, and transformation of data unless noted otherwise in this documentation.

High-speed Oracle GoldenGate replication can be used to refresh a Teradata cache environment with minimal latency. In addition, with its heterogeneous support, Oracle GoldenGate enables the Teradata data store to be used as a data integration point for other data sources.

10 Understanding What's Supported for Teradata

This chapter contains information on database and table features supported by Oracle GoldenGate.

Topics:

- Supported Teradata Data Types
- Supported Objects and Operations for Teradata
- Non-Supported Operations for Teradata

10.1 Supported Teradata Data Types

The following table shows the Teradata data types that Oracle GoldenGate supports. Any limitations or conditions that apply follow this table.

Data type	v15.x	v16.x
`BLOB`	Yes	Yes
`BYTEINT`	Yes	Yes
`VARBYTE`	Yes	Yes
`BIGINT`	Yes	Yes
`BYTEINT`	Yes	Yes
`DATE`	Yes	Yes
`DECIMAL - 18 and under`	Yes	Yes
`DECIMAL - 19 to 38`	Yes	Yes
`DOUBLE PRECISION`	Yes	Yes
`FLOAT`	Yes	Yes
`INTEGER`	Yes	Yes
`NUMERIC - 18 and under`	Yes	Yes
`NUMERIC - 19 to 38`	Yes	Yes
`REAL`	Yes	Yes
`SMALLIINT`	Yes	Yes
`TIME`	Yes	Yes
`TIMESTAMP`	Yes	Yes
`INTERVAL`	Yes	Yes
`INTERVAL DAY`	Yes	Yes

Data type	v15.x	v16.x
`INTERVAL DAY TO HOUR`	Yes	Yes
`INTERVAL DAY TO MINUTE`	Yes	Yes
`INTERVAL DAY TO SECOND`	Yes	Yes
`INTERVAL HOUR`	Yes	Yes
`INTERVAL HOUR TO MINUTE`	Yes	Yes
`INTERVAL HOUR TO SECOND`	Yes	Yes
`INTERVAL MINUTE`	Yes	Yes
`INTERVAL MINUTE TO SECOND`	Yes	Yes
`INTERVAL MONTH`	Yes	Yes
`INTERVAL SECOND`	Yes	Yes
`INTERVAL YEAR`	Yes	Yes
`INTERVAL YEAR TO MONTH`	Yes	Yes
`CHAR`	Yes	Yes
`CLOB`	Yes	Yes
`CHAR VARYING`	Yes	Yes
`LONG VARCHAR`	Yes	Yes
`VARCHAR`	Yes	Yes
`GRAPHIC`	Yes	Yes
`LONG VARGRAPHIC`	Yes	Yes
`VARGRAPHIC`	Yes	Yes
`PERIOD (DATE)`	Yes	Yes
`PERIOD (TIME)`	Yes	Yes
`PERIOD (TIMESTAMP)`	Yes	Yes
`UDT`	Yes	Yes

- Limitations of Support for Numeric Data Types
- Limitations of Support for Single-byte Character Data Types
- Conditions and Limitations of Support for Multi-byte Character Data
- Limitations of Support for Binary Data Types
- Limitations of Support for Large Object Data Types
- Limitations of Support for Date Data Types
- Limitations of Support for IDENTITY Data Types

10.1.1 Limitations of Support for Numeric Data Types

When replicating these data types from a different type of database to Teradata, truncation can occur if the source database supports a higher precision that Teradata does.

The support of range and precision for floating-point numbers depends on the host machine. In general, the precision is accurate to 16 significant digits, but you should review the database documentation to determine the expected approximations. Oracle GoldenGate rounds or truncates values that exceed the supported precision.

10.1.2 Limitations of Support for Single-byte Character Data Types

Single-byte character types are fully supported within a single-byte Latin character set between other databases and Teradata. A `VARCHAR` or `CHAR` column cannot have more than 32k-1 bytes. If using UTF-16, this is 16k-2 characters.

10.1.3 Conditions and Limitations of Support for Multi-byte Character Data

Conditions and limitations of support for multi-byte character data are as follows:

- Install Oracle GoldenGate on a Windows or Linux replication server.
- Use the Teradata ODBC driver version 12.0.0.x or later.
- Do not use filtering, mapping, and transformation for multi-byte data types.
- A `CHAR` or `VARCHAR` column cannot contain more than 32k-1 bytes. If using UTF-16, these columns cannot contain more than 16k-2 characters.
- Set the ODBC driver to the UTF-16 character set in the initialization file.
- When creating Replicat groups, use the `NODBCHECKPOINT` option with the `ADD REPLICAT` command. The Replicat database checkpointing feature does not support an ODBC driver that is set to the UTF-16 character set. Checkpoints will be maintained in the checkpoint file on disk.

10.1.4 Limitations of Support for Binary Data Types

No limitations. These data types are supported between other source databases and Teradata targets.

10.1.5 Limitations of Support for Large Object Data Types

The following are limitations of support for large object data types.

- To replicate large objects from other databases to Teradata, use Teradata ODBC driver version 12.0 or higher on the target system. The target must support large objects that are delivered by ODBC.
- Enable the `UseNativeLOBSupport` flag in the ODBC configuration file. See the Teradata ODBC documentation.

10.1.6 Limitations of Support for Date Data Types

The following are limitations of support for date data types:

- `DATE`, `TIME`, and `TIMESTAMP` are fully supported when replicated from a different type of source database to Teradata.

- `TIME with TIMESZONE`, `TIMESTAMP with TIMEZONE`, and `INTERVAL` are not supported from a different type of source database to Teradata.
- Oracle GoldenGate supports timestamp data from 0001/01/03:00:00:00 to 9999/12/31:23:59:59. If a timestamp is converted from GMT to local time, these limits also apply to the resulting timestamp. Depending on the timezone, conversion may add or subtract hours, which can cause the timestamp to exceed the lower or upper supported limit.
- Oracle GoldenGate does not support negative dates.

10.1.7 Limitations of Support for IDENTITY Data Types

`IDENTITY` must be configured as `GENERATED BY DEFAULT AS IDENTITY` on the target to enable the correct value to be inserted by Replicat.

10.2 Supported Objects and Operations for Teradata

This section lists the data operations and database objects that Oracle GoldenGate supports.

- Oracle GoldenGate supports the maximum number of columns per table that is supported by the database.
- Truncating operations are supported with the use of the `GETTRUNCATES` parameter with Oracle GoldenGate 12.2.*x* and greater.
- Limitations on Automatic Heartbeat Table support are as follows:
  - The `ALTER HEARTBEATTABLE` command is not supported and if used is ignored.
  - The `ADD HEARTBEATTABLE` command with the `FREQUENCY`, `PURGE_FREQUENCY`, or `RETENTION_TIME` option is not supported. When any of these options are specified with the `ADD HEARTBEATTABLE` command, a warning is displayed that the option is ignored.
  - Since Teradata does not have any internal event/job schedulers, automatic purging of heartbeat history data does not occur. You need to explicitly delete or truncate records periodically from the heartbeat history table.

10.3 Non-Supported Operations for Teradata

This section lists the data operations that Oracle GoldenGate does not support.

- Extract (capture)
- DDL

11 Preparing the System for Oracle GoldenGate

This chapter contains guidelines for preparing the database and the system to support Oracle GoldenGate. This chapter contains the following sections:

Topics:

- Preparing Tables for Processing

11.1 Preparing Tables for Processing

The following table attributes must be addressed in an Oracle GoldenGate environment.

- Disabling Triggers and Cascade Constraints
- Assigning Row Identifiers

11.1.1 Disabling Triggers and Cascade Constraints

Disable triggers, cascade delete constraints, and cascade update constraints on target Teradata tables. Oracle GoldenGate replicates DML that results from a trigger or cascade constraint. If the same trigger or constraint gets activated on the target table, it becomes redundant because of the replicated version, and the database returns an error. Consider the following example, where the source tables are `emp_src` and `salary_src` and the target tables are `emp_targ` and `salary_targ`.

1. A delete is issued for `emp_src`.
2. It cascades a delete to `salary_src`.
3. Oracle GoldenGate sends both deletes to the target.
4. The parent delete arrives first and is applied to `emp_targ`.
5. The parent delete cascades a delete to `salary_targ`.
6. The cascaded delete from `salary_src` is applied to `salary_targ`.
7. The row cannot be located because it was already deleted in step 5.

11.1.2 Assigning Row Identifiers

Oracle GoldenGate requires unique row identifiers on the source and target tables to locate the correct target rows for replicated updates and deletes. Source tables can have any kind of key listed in How Oracle GoldenGate Determines the Kind of Row Identifier to Use, except for tables of a SQL Server Standard Edition instance, which require a primary key. If there is no primary key identified on a table that has fixed-length columns, the length of one of the fixed-length columns must be below 3800 bytes.

11.1.2.1 How Oracle GoldenGate Determines the Kind of Row Identifier to Use

Unless a `KEYCOLS` clause is used in the `TABLE` or `MAP` statement, Oracle GoldenGate selects a row identifier to use in the following order of priority:

1. Primary key (required for tables of a Standard Edition instance).
2. First unique key alphanumerically that does not contain a timestamp or non-materialized computed column.
3. If neither of these key types exist , Oracle GoldenGate constructs a pseudokey of all columns that the database allows to be used in a unique key, excluding those that are not supported by Oracle GoldenGate in a key or those that are excluded from the Oracle GoldenGate configuration. For SQL Server, Oracle GoldenGate requires the row data in target tables that do not have a primary key to be less than 8000 bytes.

> **Note:**
>
> If there are types of keys on a table or if there are no keys at all on a table, Oracle GoldenGate logs a message to the report file. Constructing a key from all of the columns impedes the performance of Oracle GoldenGate on the source system. On the target, this key causes Replicat to use a larger, less efficient `WHERE` clause.

11.1.2.2 Using KEYCOLS to Specify a Custom Key

If a table does not have an applicable row identifier, or if you prefer that identifiers are not used, you can define a substitute key, providing that the table has columns that always contain unique values. You define this substitute key by including a `KEYCOLS` clause within the Extract `TABLE` parameter and the Replicat `MAP` parameter. The specified key overrides any existing primary or unique key that Oracle GoldenGate finds.

12 Configuring Oracle GoldenGate

This chapter describes how to configure Oracle GoldenGate Replicat. This chapter contains the following sections:
Topics:

12.1 Configuring Oracle GoldenGate Replicat

This section highlights the basic Replicat parameters that are required for most target database types. Additional parameters may be required, see the Oracle GoldenGate installation and configuration documentation for your target database and the *Reference for Oracle GoldenGate*, .

Perform these steps on the target replication server or target database system.

1. Configure the Manager process according to the instructions in *Administering Oracle GoldenGate*.
2. In the Manager parameter file, use the `PURGEOLDEXTRACTS` parameter to control the purging of files from the local trail.
3. Create a Replicat checkpoint table. There are multiple options for this purpose, see *Administering Oracle GoldenGate*.
4. Create a Replicat group. For documentation purposes, this group is called `rep`.

   ```
   ADD REPLICAT rep, EXTTRAIL remote_trail
   ```

   Use the `EXTTRAIL` argument to link the Replicat group to the remote trail that you specified for the data pump on the source server.
5. Use the `EDIT PARAMS` command to create a parameter file for the Replicat group. Include the parameters shown in Example 12-1 plus any others that apply to your database environment.

Example 12-1 Parameters for the Replicat Group

```
-- Identify the Replicat group:
REPLICAT rep
-- Specify database login information as needed for the database:
[TARGETDB dsn2,] [USERID user id[, PASSWORD pw]]
-- Specify error handling rules (See the NOTE following parameter file):
REPERROR (error, response)
-- Specify tables for delivery:
MAP owner.table, TARGET owner.table;
```

Note:

In a recovery situation, it is possible that Replicat could attempt to apply some updates twice. If a multiset table is affected, this could result in duplicate rows being created. Use the `REPERROR` parameter in the Replicat parameter file so that Replicat ignores duplicate rows.

12.2 Additional Oracle GoldenGate Configuration Guidelines

The following are additional considerations to make once you have installed and configured your Oracle GoldenGate environment.

- Handling Massive Update and Delete Operations
- Preventing Multiple Connections
- Performing Initial Synchronization

12.2.1 Handling Massive Update and Delete Operations

Operations that update or delete a large number of rows will generate discrete updates and deletes for each row on the subscriber database. This could cause a lock manager overflow on the Teradata subscriber system, and thus terminate the Replicat process.

To avoid these errors, temporarily suspend replication for these operations and then perform them manually on the source and target systems. To suspend replication, use the following command, which suspends replication for that session only. The operations of other sessions on that table are replicated normally.

```
set session override replication on;

commit;
```

12.2.2 Preventing Multiple Connections

By default, the Replicat processes create a new connection for catalog queries. You can prevent this extra connection by using the `DBOPTIONS` parameter with the `NOCATALOGCONNECT` option.

12.2.3 Performing Initial Synchronization

Perform an initial synchronization of the source and target data before using Oracle GoldenGate to transmit transactional changes for the first time to configure an initial load, see *Administering Oracle GoldenGate*.

13 Common Maintenance Tasks

This chapter contains instructions for performing some common maintenance tasks when using the Oracle GoldenGate replication solution.

Topics:

- Modifying Columns of a Table

13.1 Modifying Columns of a Table

To modify columns of a table:

1. Suspend activity on the source database for all tables that are linked to Oracle GoldenGate.
2. Start GGSCI.
3. In GGSCI, issue this command for the Replicat group:

   ```
   INFO REPLICAT group
   ```

4. On the `Checkpoint Lag` line, verify whether there is any Replicat lag. If needed, continue to issue `INFO REPLICAT` until lag is zero, which indicates that all of the data in the trail has been processed.
5. Stop the Replicat group.

   ```
   STOP REPLICAT group
   ```

6. Perform the table modifications on the target databases.
7. Start the Replicat process.

   ```
   START REPLICAT group
   ```

8. Allow user activity to resume on all of the source tables that are linked to Oracle GoldenGate.