

Overview

The objective of this chapter is to cover Oracle GoldenGate fundamentals, focusing on use cases and applications, concept and architecture, topologies and designs, components and building blocks and end-to-end illustrative setup for a headstart. For Oracle GoldenGate knowledgeable users, the chapter intended for refreshing, for users new to Oracle GoldenGate, it builds the foundation to continue the remaining chapters. Despite the fact that the illustrative setup and commands are developed on Oracle GoldenGate 12c, they should apply to Oracle GoldenGate supported releases, unless noted otherwise.

The chapter discusses and exemplify the following topics:

- Oracle database replication evolution and history
- Supported platforms and product set
- Enterprise applications and use cases
- Network topologies and designs
- Building blocks, architecture and processes workflow
- Command-line utilities and obey files
- Parameter files and macros
- Manager process
- Primary and secondary extracts groups for change data capture
- Change data delivery group
- Trails and trail file format
- Commit change number, checkpoint file and checkpoint table
- The data server collector
- Intermediate systems and integrated capture
- Data selection, filtering, transformation and column mapping
- Data definition file utility
- Built-in functions and tokens
- Call-out to database stored procedures and C programs
- Post deployment configuration options
- Configure primary extract for Automatic storage environment (ASM)

- Implementing high-performance parallel processing
- Generate reports and lagging
- Configure extract and replicate processes for pluggable database environment
- Demonstrating three-step configuration

The aim of the book remains, an in-depth dive into Oracle GoldenGate 12c applications for area-specific solutions presented on subsequent chapters. Even though Oracle GoldenGate concepts applies across supported platforms, the fundamentals chapter discusses Oracle GoldenGate 12c for Oracle Database 11g, Oracle Database 12c and Microsoft SQL Server 2008.

For training purposes only, an Oracle VirtualBox and VM player machines are made available, the machines provide a quick setup to practice Oracle GoldenGate 12c. Visit <http://www.oracle-ggs.com> for downloading updated virtual machines and user's guide for setup instructions. Terms and conditions applied.

Oracle GoldenGate, an enterprise software

For years, Oracle has been one of the world largest enterprise software company, now Oracle software found globally at fortune companies running core business applications. These companies continue to have real demands for solutions to optimize, consolidate and integrate business functions across the enterprise by running heterogeneous core business applications systems. Oracle GoldenGate is part of Oracle data integration solution (DIS) portfolio, when deployed as best practices, it becomes the platform for data consolidation and integration delivering best performance, flexible architecture, reliability and extendibility across the enterprise.

As Oracle continues the research and development for the world leading object relational database management system (ORDBMS), enterprise applications demand more integration among database platforms, and when it's for the enterprise, the integration must be *heterogeneous*, and that is a key differentiator to stand out and stay competitive within the data integration market. Oracle GoldenGate optimized to run on popular database management systems such as Oracle Database, Oracle MySQL, Microsoft SQL Server, IBM DB2, etc.

Enterprise applications require moving transactional data in real-time among systems hosted globally, it has been traditionally processed using in-house developed applications utilizing database links, triggers and materialized views among the enterprise databases, and when it's for the enterprise, data growth maintains a linear pattern and mostly requires online data movement with zero refresh window timeframe. Another key differentiator, Oracle GoldenGate enables real-time data movement with *sub-second latency* using an optimized change data capture and delivery

processes, while employing basic-to-complex data transformation and consolidation for data in transit.

Performance and data consistency cannot be compromised at the enterprise level. Oracle GoldenGate uses its proprietary technique to maintain data consistency across systems regardless of databases type. Using an in-memory processing, Oracle GoldenGate treats a transaction as a complete logical unit of work, when the transaction committed, it's assigned an Oracle GoldenGate commit sequence number (CSN) or discarded when the transaction is rolled back. Committed transactions are moved to the next downstream process. At the target system, the transactions reassembled and applied with the same capture order for maintaining *transaction integrity*. By having Oracle GoldenGate captures committed transactions only, this has a substantial performance enhancement on high-volume transactional systems.

And, that explains Oracle GoldenGate in terms of *heterogeneous* support, *sub-second latency* for data movement and transactional data *integrity*, making it an enterprise software for powering critical mission applications.

Enterprises running critical mission applications requires well engineered software, promoting flexible and extendable architecture, and this is what is Oracle GoldenGate is all about. Implementing Oracle GoldenGate is site specific, based on that site business and technical requirements, a flexible dynamic architecture is developed and deployed. Based on the network topology, capture and delivery groups may be designed loosely or tightly coupled, for efficient resource utilization with varying degree of parallelism and custom development for unlimited extendibility.

Because Oracle GoldenGate uses its own checkpoint sub-system, it enabled an optimized restart due to any type of failures. Extract and replicate processes continuously checkpoint their read and write positions. Data security regulation mandates data availability at all times, Oracle customers have been using Oracle Data Guard to meet data security requirements for disaster recovery, Oracle GoldenGate extract groups support capture from standby archive log file, this integration seamlessly promotes data movement from the standby database leaving the source production database dedicated for online transactions, an added advantage for report offloading instance from a standby database environment.

Oracle data replication evolution

Data replication has been always common requirement for database applications. Transparent data replication has been part of Oracle database framework. Hence, Oracle database replication developed using PL/SQL database application programmatic interface (API) and enabled from within PL/SQL or Oracle Enterprise

Manager (GUI). For high-volume transactions, handling data replication outside of the database contributes to achieve real-time data replication across heterogeneous systems as its engineered by Oracle GoldenGate.

The following describes the history of Oracle database replication from inception to current, that is from Oracle 7 to Oracle GoldenGate 12c.

Oracle replication, June 1992

Oracle replication introduced with Oracle 7.0. It was simple infrastructure, database triggers are associated with database tables, moving data to remote database using database links. Subsequently, the feature has been completely redeveloped as Oracle advanced replication.

Oracle advanced replication, May 1994

Oracle 7.1 introduced Oracle advanced replication. A complete data replication framework for creating complex database replication network, such as master-to-master replication. Using PL/SQL API or Oracle Enterprise Manager (GUI), data replication API infrastructure generated and enabled for data replication. Basic predefined data conflict resolutions defined and handled internally.

Oracle Streams, May 2002

Oracle Database 9.2 introduced Oracle Streams. Streams uses advance queuing technology for data capture, propagate and apply workflow. Oracle Streams supports creating complex data replication network, such as master-to-master by using PL/SQL API or Oracle enterprise manager (GUI). Oracle Streams has superseded Oracle advanced replication.

Oracle GoldenGate, September 2009

It's data replication and beyond. Oracle GoldenGate enables high availability solutions, real-time data integration, transactional change data capture (CDC) and delivery, high-performance data replication and complex data transformations. Oracle GoldenGate engineered for high-performance real-time heterogeneous data replication, supporting the leading database management systems with three key differentiators: heterogeneous, sub-second latency and data integrity.

Oracle strategy for data replication solutions stated Oracle database Streams is superseded by Oracle GoldenGate as quoted on Oracle Statement of Direction, November 2012.

“Oracle Streams continues to be supported but will not be enhanced to support any new database features in future releases. This means that Streams will continue to support Exactly what it supports today (11.2) in future releases. It also means that any new database features, data types, or capabilities will not be supported by Streams. Customers who want to take advantage of the new data types and features will need to license Oracle GoldenGate.” – Oracle Corporation

With that said, the direction definitely strategic and positively reflects the continued research and development efforts for more comprehensive Oracle GoldenGate product set, and highly integrated with other related Oracle products and most of all Oracle Data Integrator (ODI) and Oracle Database 12c data guard technology.

What is new for Oracle GoldenGate 12c

Oracle GoldenGate 12c version 12.1.2.0.0 was made available on October 2013, the following is a brief description of the new features and the enhancements introduced by Oracle GoldenGate 12c.

Database support

Oracle GoldenGate 12c supports Oracle Database 12c, the database for cloud computing. Oracle GoldenGate continues supporting and enhancing existing non-Oracle database platforms, MS SQL Server, Teradata, mainframe and external adapters.

Software installation

Oracle GoldenGate 12c is shipped with graphical user interface (GUI) installer, supporting Oracle Database 11g and Oracle Database 12c. The 5-step easy-to-use wizard de-supported the manual installation of previous releases. The GUI installer installs the software, creates the directories structure and starts the manager process. The software is bundled for Oracle Database 11g and Oracle Database 12c.

Integrated capture

For users who want to isolate Oracle GoldenGate capture process from the production database system rely on integrated capture extracts. With integrated capture, Oracle GoldenGate 12c installed on a downstream system continuously

receives REDO log files from the source system. Oracle GoldenGate 12c has transparently enhanced the performance of integrated capture and replicate processes. Integrated capture will be the area of enhancements on future releases.

Oracle multitenant container database

Oracle GoldenGate 12c supports capturing from and delivering to Oracle multitenant container database (CDB). A primary extract captures from a pluggable database (PDB), writes data to a local trail, and delivers to a remote pluggable database.

Extended TABLE and MAP statement

Oracle GoldenGate 12c support three-part object name for the TABLE and MAP statement. The object name includes <container>.<schema>.<table name>.

Enhanced DDL support

DDL replication does not need database triggers setup. Oracle GoldenGate 12c performs native DDL replication. This is the default mode, switching back to capture DDL mode requires Oracle GoldenGate 12c DDL setup.

Character set

Oracle GoldenGate 12c enhanced the change data delivery by providing automatic character set support, eliminating setting NLS_LANG parameter on the replicate process.

Initial-load data types

Oracle GoldenGate 12c initial-load remote task supports all Oracle data types including BLOB, CLOB, NCLOB, LONG, UDT and XML. Making direct initial-load method an excellent technique.

MS SQL Server

Oracle GoldenGate 12c supports compressed backup, in addition to the previously supported uncompressed backup. Compressed backup uses less I/O therefore increasing the backup speed.

Oracle GoldenGate line of products

Oracle GoldenGate 12c bundles several Oracle software licensable separately. The following is a brief description of Oracle GoldenGate product set. A major software component of Oracle GoldenGate is adapters, used to support legacy systems and site-specific data processing requirements. Adapters mostly adapted by financial institutes using BASE24 system for payment processing. IBM mainframe is has been always supported by Oracle GoldenGate for enabling data movement to modern systems across the enterprise.

Oracle GoldenGate product set described next, starting with Oracle GoldenGate core for basic data capture and delivery.

Oracle GoldenGate core

The core is made of Oracle GoldenGate capture and Oracle GoldenGate apply. Oracle GoldenGate core is classified as:

- Oracle GoldenGate for Oracle Database
- Oracle GoldenGate for non-Oracle database
- Oracle GoldenGate for mainframe

Oracle GoldenGate core enables real-time change data capture (CDC), moving high volumes of transactional data between databases with very low latency and data integrity using combinations of heterogeneous source and target databases. The separation of capture and apply Oracle GoldenGate core license enables faster return on investment (ROI) and cost-effective integration with non-Oracle data integration software.

An added advantage of Oracle GoldenGate core is support for Oracle Database standard edition, providing high availability solutions for non-enterprise users using Oracle database standard edition. Oracle GoldenGate core includes full use of XStream and Active Data Guard. XStream is an API for enabling client applications to receive and send real-time data, Active Data Guard enables the physical standby database to be opened for read-only while applying changes from the primary database. Both XStream and Active Data Guard require running Oracle database enterprise edition.

Two essential components of Oracle GoldenGate core are GoldenGate Software Command Interface (GGSCI) and the LOGDUMP utilities. Use GGSCI to define configure, create and deploy GoldenGate components from the command-line, use

LOGDUMP utility for reading, searching and managing trail files and troubleshooting. Refer to chapter 13 for details regarding LOGDUMP.

Management pack for Oracle GoldenGate

The management pack composed of a set of Oracle GoldenGate products providing a graphical user interface (GUI) for managing Oracle GoldenGate using the latest Java and Web-base technologies.

Oracle GoldenGate director

A graphical user interface (GUI) for defining, configuring, managing, monitoring and reporting on Oracle GoldenGate instances through Java client and web-based interfaces. Director provides a diagramming visual tool for creating diagrams for new or existing Oracle GoldenGate instances. Using drag-and-drop technique, the diagramming tool creates capture and delivery groups, initial load, capture, trial, files and replicates easily and instantly. Figure 1-1 shows the diagramming tool.

Refer to chapter 9 for details regarding Oracle GoldenGate director.

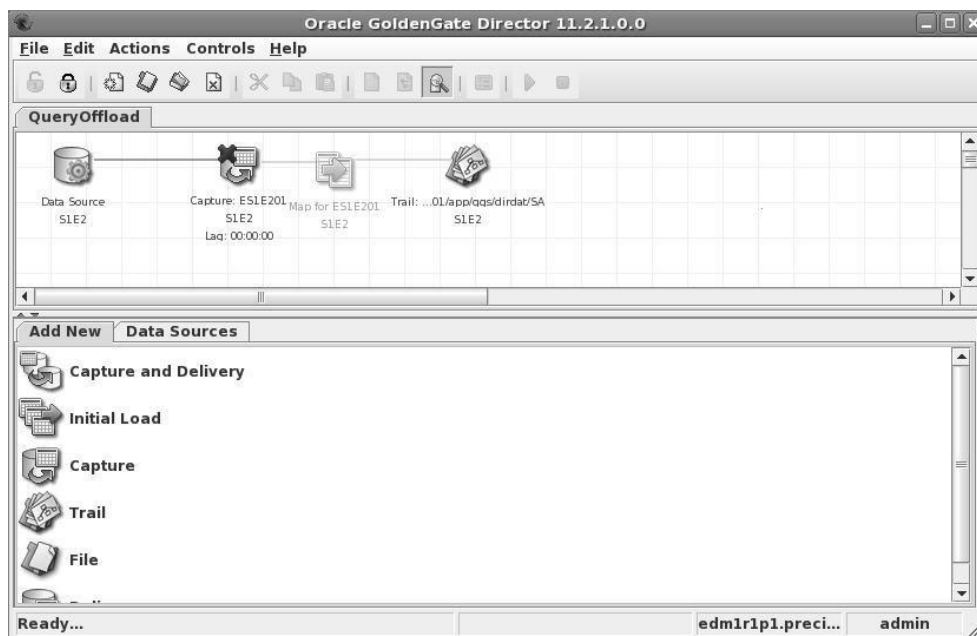


Figure 1-1: Oracle GoldenGate director visual diagramming tool

Oracle GoldenGate monitor

Oracle GoldenGate monitor is a browser-based for viewing purpose only, it does not modify Oracle GoldenGate configuration files. It's used to monitor Oracle GoldenGate instance process status, operating and performance statistics, and reports runtime events. The major feature of Oracle GoldenGate monitor is external notification using either email or SNMP.

Oracle GoldenGate Plug-in for Oracle Enterprise Manager

Oracle Enterprise Manager Cloud Control 12c management pack for Oracle GoldenGate enables managing Oracle GoldenGate instances from a centralized integrated console. Upon discovering Oracle GoldenGate instance as a target by the agent, management of Oracle GoldenGate instance delivered from within Oracle Enterprise Manager Cloud Control.

Oracle Enterprise Manager Cloud Control 12c management pack for Oracle GoldenGate provides the following functionalities:

- Centralized management of multiple Oracle GoldenGate instances
- General health and performance review

Oracle GoldenGate veridata

For sites requiring real-time data monitoring, Oracle GoldenGate Veridata provides an easy-to-use yet powerful solution for identifying out-of-synch data before becoming very diverted causing data inconsistency. Deployed together with the Oracle GoldenGate real-time data replication product or separately, Oracle GoldenGate Veridata ensures data consistency is maintained across configured two databases, playing a major rule when performing zero downtime migration and upgrade using Oracle GoldenGate. Figure 1-2 depicts Oracle GoldenGte Veridata.

The major component for configuring Oracle GoldenGate are:

- GoldenGate Veridata server
- Apache Tomcat Web server
- Apache Tomcat Web Server administration tool
- Goldengate Veridata Web
- GoldenGate Veridata repository
- GoldenGate Veridata agent

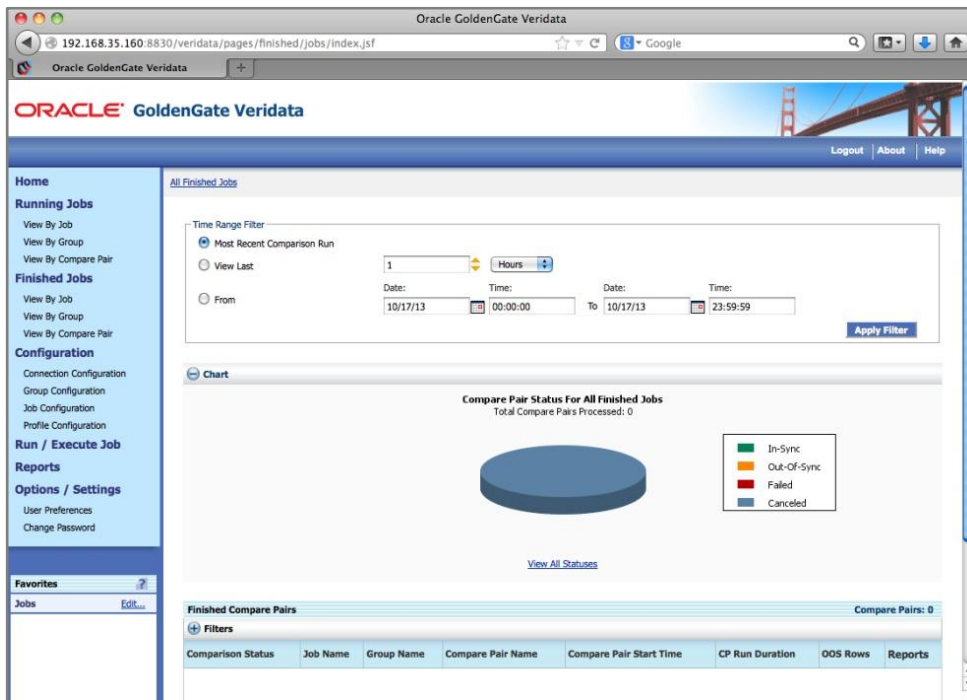


Figure 1-2: Oracle GoldenGate Veridata

Refer to chapter 6 For details regarding Oracle GoldenGate Veridata.

Oracle GoldenGate adapters

Oracle GoldenGate provide support for different types adapters, these adapters highly promote Oracle GoldenGate integration with legacy systems running legacy applications and ETL-based applications.

Oracle GoldenGate adapter for flat file

Flat file adapter enables Oracle GoldenGate to produce a series of sequential flat files data and control files from transactional data capture trail files of a source system. Widely used for data warehouse applications ETL and proprietary or legacy applications.

Oracle GoldenGate flat file adapter uses a special type of extract, implemented as user exit, to produce rolling data flat file to be consumed by an interface such as Oracle external tables. The data is formatted by Oracle GoldenGate to the specifications of

the target applications either as delimiter separated values or length delimited values. The advantage for using flat file adapter, near real-time data feeds to legacy systems is accomplished by minimizing the window elapsed timeframe for batch file rollover.

Refer to chapter 12 to learn more about flat file adapter.

Oracle GoldenGate adapter for Java

Java Message Service (JMS) messages generated from Java applications are handled by a special type of capture, VAM message capture. Using a source definitions file a local trail created and routed to the remote system. The remote system data collector server reassemble the data and create a remote trail for the replicate process.

Refer to chapter 12 to learn more about Java adapter.

Oracle GoldenGate application adapter for BASE24

BASE24 is a series of applications produced by ACI Worldwide, Inc. used by payment processing systems for transaction routing and authorization, mostly found at large financial institutions. Oracle GoldenGate provides adapters to integrate BASE24 components heterogeneously, moving data from ATM networks to the core banking systems for online processing. The adapters available for BASE24:

D24 adapter

Enables transactional data to be synchronized bi-directionally in real time with high-availability support while maintaining data consistency and integrity.

N24 adapter

Performs routing and coordination of the notification message to all BASE24 satellite processes up on the file refresh has been completed, providing transparent configuration for bidirectional and unidirectional data flow.

T24 adapter

Resolves issues when moving data among platforms and databases, acts on the transaction log of TLF and PTLF tokens for all transaction types, ensuring the correct order and format.

Oracle GoldenGate Logger for Enscribe

Provides the mechanism for change data capture on non-audited Enscribe database platforms. The Logger requires the installation of GGSLIB for binding the Oracle GoldenGate and the user's NonStop application. When an Enscribe operation (such as WRITE) executes, GGSLIB intercepts it and sends the record to Logger. The Logger writes the records to the universal format log trail, which can then be processed like any other log-based source.

Supported database platforms

Oracle GoldenGate heterogeneity aims to support major database management systems. Depending on database type, support for capture and apply is determined, and if the change data capture is log-based. Table 1-1 lists the current supported database platforms by Oracle GoldenGate 11g and higher.

Database Platform	Log-based Capture	Delivery
Oracle Database	☑	☑
Microsoft SQL Server	☑	☑
Oracle MySQL	☑	☑
IBM DB2 for Linux, UNIX and Windows	☑	☑
IBM DB2 for i	☑	☑
IBM DB2 for z/OS	☑	☑
Sybase	☑	☑
SQL/MX	☑	☑
Teradata		☑
Oracle TimesTen		☑
HP Neoview		☑
Netezza		☑
ETL products		☑
JMS message queues		☑

Table 1-1: Oracle GoldenGate supported databases

The vast majority of database capture method is log-based, translated to low-impact, high-performance on the source database system. The log-based capture technology is a key differentiator of Oracle GoldenGate, other extraction tools adapt queuing technology, highly impacting the source database system performance.

Oracle GoldenGate 12c for Oracle database supports two forms of log-based capture: classic capture and Integrated capture. Migration from classic to integrated capture is recommended as direction of new features and enhancements are toward integrated capture architecture. Refer to chapter 10 to learn more about integrated capture.

Supported operating systems

From the desktop PCs to the mainframe computers enabling the foundation for a comprehensive integration environments. Oracle GoldenGate 11g and higher support the operating systems list on table 1-2.

Operating system	Version	Hardware
AIX	5.1, 5.2 and 5.3	IBM PowerPC
HP Nonstop	D46, D47, D48, G06 and H06	NSK
HP-UX	11.11, 11.23 and 11.31	HP Intel IA64 and HP PA-RISC
Oracle Linux	3.0, 4.0, 5.0 and 6.0	AMD/Intel x64, Intel x86 and Intel IA64
RedHat AS	2.1, 3.0, 4.0 and 5.0	AMD/Intel x64, Intel x86 and Intel IA64
Solaris	8, 9 and 10	Sun SPARC
SuSE	9 and 10	Opteron and Intel x86
Tru64	5.1	HP OSF
Windows	2000, 2003, 2008 and XP	AMD/Intel x64, Intel x86 and Intel IA64
z/OS	1.04 and 1.06	Mainframe

Table 1-2: Operating systems support

Oracle GoldenGate applications

The extendibility, flexibility and adaptability of Oracle GoldenGate enables the implementation of the software across an enterprise to meet diversity of business objectives. The major areas of applying Oracle GoldenGate listed hereunder followed by a brief description and diagram.

- Query offloading techniques
- Business continuity solutions
- Zero downtime migration and upgrade
- Operational data warehouse and business intelligence
- Data integration services

Query offloading techniques

Reports are an integral part of any database application systems, the performance overhead associated with running these report on the production systems always has a negative impact on online users, the real customers. Running standard reports such as end-of-year, month closing, etc. demand more system resources and normally scheduled during less workload hours, these are relatively back-dated reports with data gap proportional to the batch window for an online systems operating 24/7 basis using different types for input interfaces.

The solution is implementing Oracle GoldenGate at the enterprise level for the purpose of queries offloading, enabling efficient resource utilization across systems with real-time change data capture (CDC) from OLTP systems acting as data sources to an adapted database instance dedicated for reporting only. Heterogeneous change data capture is very common and proven to be cost-effective, for example the data source OLTP systems running on Oracle Database 11g for Solaris 11 and the reporting instance running on MySQL 5 on Oracle Enterprise Linux 6.

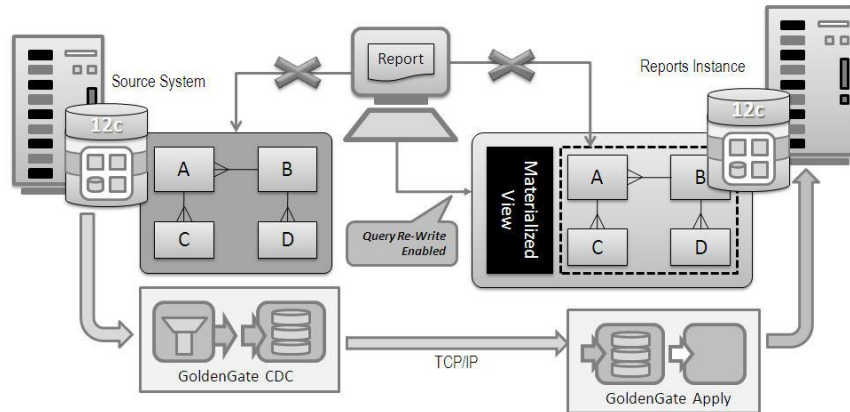


Figure 1-3: Query offloading technique

Oracle database and MS SQL Server shipped with features for optimizing out-of-box report applications. Oracle database materialized view enables query rewrite which greatly enhances report runtime. Figure 1-3 depicts a high-level design for query offloading using Oracle GoldenGate.

Refer to chapter 4 to learn how to implement Oracle GoldenGate for query offloading.

Business continuity solutions

Oracle GoldenGate capable of delivering different architectures for business continuity solutions, where each solution implemented to suite specific high availability needs, maximize resources utilization and integration with existing high availability solutions such as Oracle Real Application Clusters (RAC) and Oracle Data Guard (DG).

In general, Oracle GoldenGate high availability solutions are divided into two areas: Disaster recovery and high availability.

Disaster recovery

The remote database site remains on alert for disaster recovery situation. Oracle GoldenGate unidirectional configuration delivers real-time change data capture (CDC) and change data delivery ensuring the gap between the primary (source) and the standby (target) databases is very minimum, with sub-seconds latency. In the event of primary site failure, the standby database is on read-write mode ready to accept connections. Using Oracle GoldenGate Veridata is recommended for maintaining consistent view of the two databases for detecting and applying transactions causing data divergence.

Oracle GoldenGate for disaster recovery is a live-standby database. The database is open for read-write operations, unlike Oracle data guard where the database is either on recovery mode or on read-only recovery mode combined.

High availability

When using Oracle GoldenGate for high availability solution, it applies the fault-tolerance concept system-wide, which includes the database instance and the database files, this is achieved by implementing Oracle Real Application Clusters (RAC) and Oracle Data Guard (DG) combined. And, because the local and the remote Oracle GoldenGate instances are connected via TCP/IP, constraints related to distance between the two systems are eliminated.

Figure 1-4 shows the high availability architecture is achieved by implementing Oracle GoldenGate bidirectional configuration, users connect to either system based on application, location, user type, etc., providing application scalability and best resource utilization. Data conflict is a major concern and should be carefully addressed when using Oracle GoldenGate for high availability active-active configuration. Oracle GoldenGate built-in data conflict resolution routines minimized custom development and maintenance of data conflict handling.

However; Oracle GoldenGate does not eliminate Oracle Real Application Clusters (RAC) or Oracle Data Guard (DG), to the contrary it creates a first-class high availability platform. Oracle GoldenGate takes advantage of RAC by registering the manager processes (the application) as a resource in Oracle Clusterware. This integration ensures that Oracle GoldenGate is highly available in the event of a node failure. The Oracle GoldenGate instance transparently migrated to a surviving node. Oracle GoldenGate takes advantage of Oracle Data Guard by performing change data capture from the standby database archive log, leaving the primary site intact, which provide the foundation for near real-time reporting instance, offloading the source database from long running queries and reports.

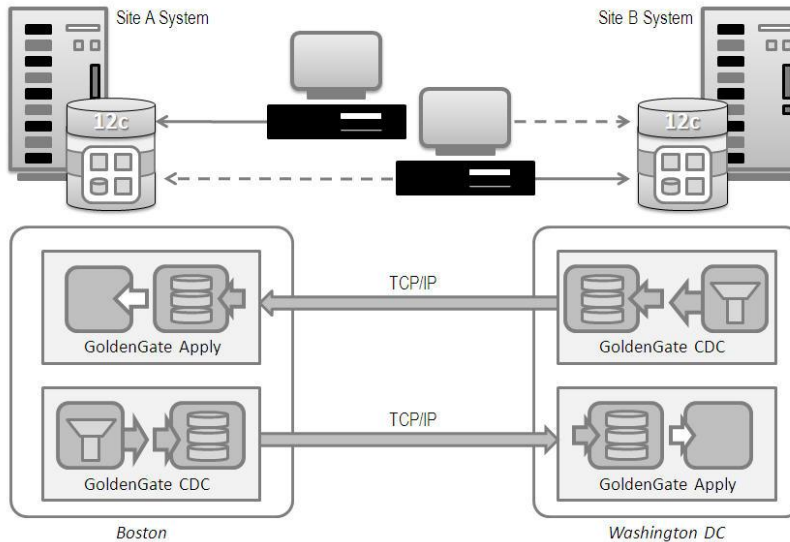


Figure 1-4: Oracle GoldenGate for high-availability

Regardless of Oracle GoldenGate configurations for high availability, the target database remains open for read-write operations, it's required by Oracle GoldenGate delivery processes and other applications. Another added-value when using Oracle GoldenGate for high availability solutions is heterogeneity, the source and target systems may differ in terms of hardware architecture and database software.

Refer to chapter 5 to learn how to implement Oracle GoldenGate for high availability.

Zero downtime migration and upgrade

Oracle GoldenGate can play major rule for the migration and upgrade projects. For systems running critical mission applications, implementing Oracle GoldenGate for zero downtime works equally for homogenous and heterogeneous environment. For example, migration of Oracle Database 11g on IBM AIX (big endian) to Oracle Database 12c on Oracle Enterprise Linux (small endian), Oracle GoldenGate handles endiannes differences internally. When upgrading Oracle Database 11g to Oracle Database 12c, the Oracle Database Upgrade Assistance (DBUA) steps requires database shutdown. The downtime timeframe due to database upgrade varies, depending on different factors, but mainly the database option to upgrade. However, the most obvious benefit when using Oracle GoldenGate for migration and upgrade

is the provision of solid fail back option. The source database remains intact and active until it's time to be shutdown and decommissioned.

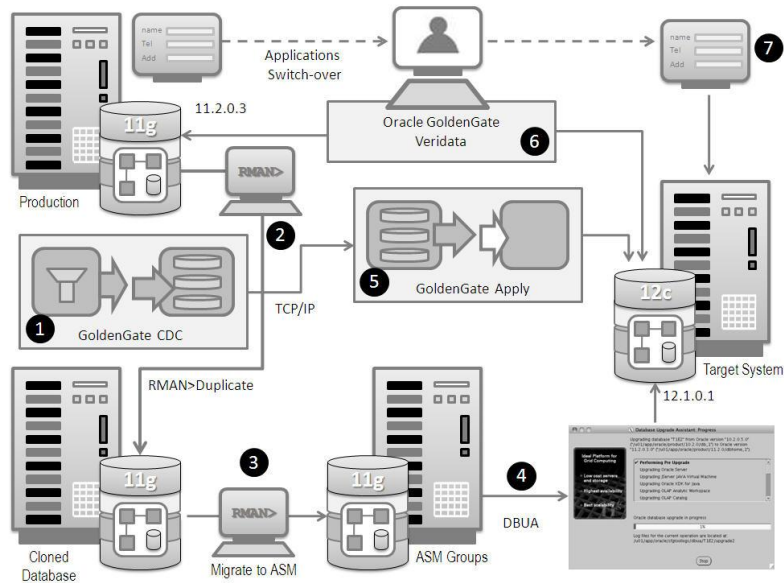


Figure 1-5: Zero downtime migration and upgrade workflow

Achieving zero downtime for migration and upgrade requires cloning the current database. The cloned database is upgraded and ultimately synchronized with the existing production database using Oracle GoldenGate. Use of Oracle GoldenGate Veridata is very recommended during the migration and upgrade. Figure 1-5 shows the steps and workflow for performing zero downtime migration and upgrade.

Refer to chapter 7 to learn how to implement Oracle GoldenGate for zero downtime migration and upgrade.

Data warehouse and operational business intelligence

Very large database (VLDB) for data warehouse and business intelligence needs optimized data feed, mostly delivered as batch windows ETL using in-house developed workflow processes. When Oracle GoldenGate implemented for online data feed to a data warehouse instance, it delivers an operational repository available online for data warehouse, and with Oracle GoldenGate data transformation and mapping capabilities, the repository is immediately available for business intelligence applications. Oracle GoldenGate eliminates the needs for refresh batch windows and customized ETL utilities. Oracle GoldenGate parallel dynamic processing feature

further enhances data feed speed across heterogeneous systems. The Integration of Oracle GoldenGate with Oracle Data Integrator (ODI) exponentially expedites and optimizes the data feed as an E-LT, promoting highly flexible system. Figure 1-6 shows Oracle GoldenGate for data warehouse.

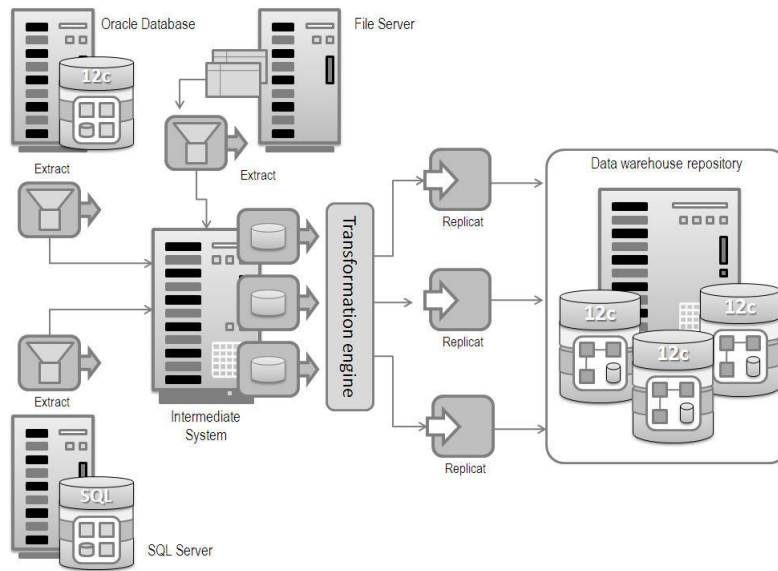


Figure 1-6: Data warehouse feed from heterogamous systems

Refer to chapter 10 to learn how to implement Oracle GoldenGate for data warehouse and business intelligence.

Data integration services

Oracle GoldenGate provides different types of adapters. These adapters enable integration of modern Java technologies and legacy systems applications with Oracle and non-Oracle database. A typical example is Oracle GoldenGate BASE24 adapters used by ATM networks for sending data to payment authorization servers . Another example is a smart-phone application sending decoded messages to an application service provider (ASP), where Oracle GoldenGate implemented as a data hub for filtering and routing data across TCP/IP to consumers. Figure 1-7 depicts payment and ATM networks for transactions processing.

Financial institutes processing payments in real-time require access to heterogeneous systems for authorization, validation and reflect changes. With Oracle GoldenGate a unified infrastructure minimizes the complexity for interfacing foreign systems and

applications. Leaving each organization with its own existing databases and platform maximizes the return of investment (ROI) uncompromised.

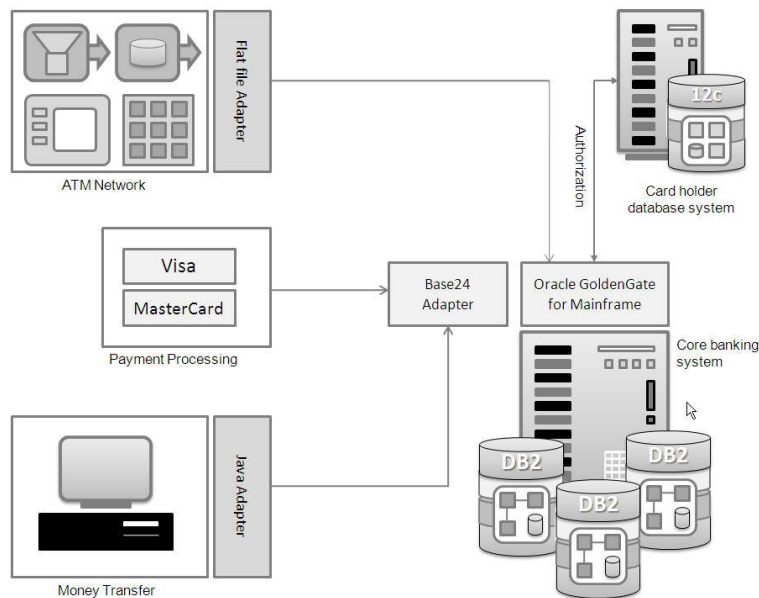


Figure 1-7: payment processing systems

Refer to chapter 12 to learn how to implement Oracle GoldenGate for data integration services.

Oracle GoldenGate building blocks

The building blocks of Oracle GoldenGate allows high degree for design flexibility for scalable performance, delivering basic-to-complex transactional data replication with transformation capabilities. The basic configuration refer to unidirectional, one-way, data replication from the source to the target database achieving real-time data movement with sub-second latency. However, Oracle GoldenGate commonly deployed to enable integration among heterogeneous systems, moving and consolidating data from various source systems to remote systems with data integrity and consistency.

Oracle GoldenGate extract groups run as loosely or tightly coupled. The primary extract processes running on source systems for change data capture (CDC), the secondary extract processes running on an intermediate systems for transformation and routing change data capture to multiple target systems. The replicate processes

are running on target systems, with parallel processing support for achieving high-performance. Figure 1-8 shows the building blocks found on the source and target systems for configuring an Oracle GoldenGate instance.

An Oracle GoldenGate instance configured for online synchronization consists of the components:

- Source and target databases
- Manager process
- Online extract process
- Local trail files
- Data pump extract process
- TCP/IP Network
- Data server collector
- Remote trail files
- Online replicate process

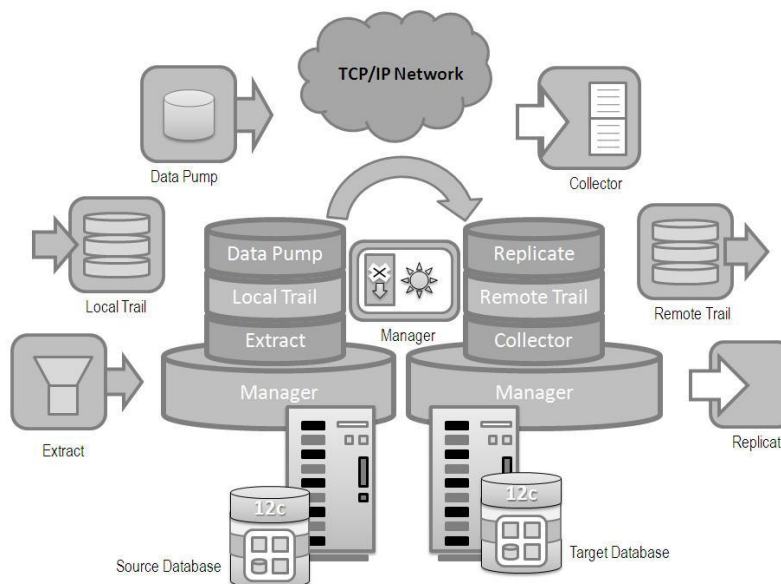


Figure 1-8: Oracle GoldenGate building blocks

The functionality for each of the building block components is described next. Figure 1-9 illustrates a simplified configuration approach, showing the components for performing initial-load and data synchronization. The initial-load reads from the source database tables, the data synchronization uses the transaction logs for online data capture and delivery. The use of data pump extract is optional but highly

recommend for supporting fault-tolerance and non-interruptible online capture in the event of network failure and avoiding redundant processing by the online extract.

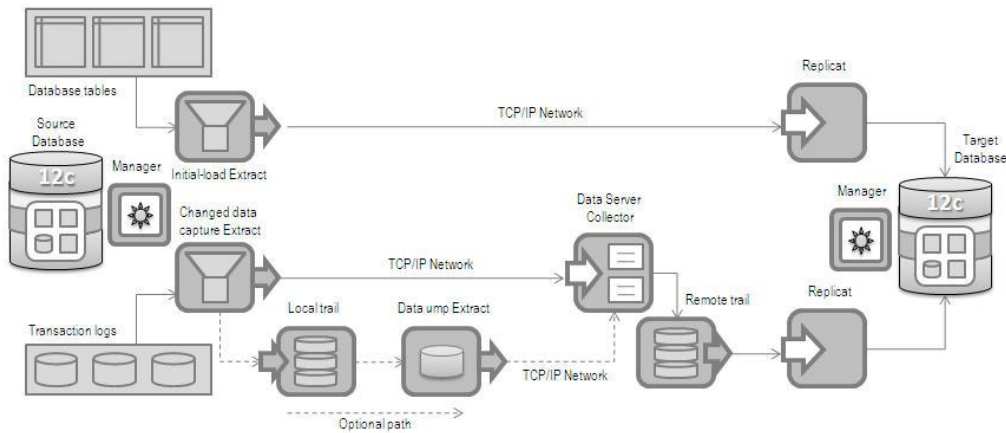


Figure 1-9: Building blocks connected

For initial-load, an initial-load extract read from the source database table and sends insert-type transaction to the remote replicate, which use the database interface to apply the transactions. Refer to chapter 3 to learn how to perform initial-load.

For change synchronization, a primary extract captures all types of transactions (DML and DDL), directly sending data to data server collector on the remote system, which in turn re-assemble the data and writes to the remote trail in a conical file format. The replicate process in the remote system, reads the trails, creates transaction units, and apply them to the target database in the same commit order occurred in the source database, ensuring data integrity and consistency.

The dotted arrows show the option of using what is called data pump extract, which reads the local trails created by the primary extract process and send them to the data server collector. This configuration avoid redundant processing in case of network failure between the source and target systems. Additionally, it offloads the primary extract from performing non-related task of routing to remote systems.

Source and target databases

Oracle GoldenGate heterogeneous support enables moving data among dissimilar source and target databases. However; The source database must be configured to meet Oracle GoldenGate operating requirements, which is database type specific.

Also, the database platform architecture must be compatible as Oracle GoldenGate 12c is available for 64-bit and 32-bit architecture.

For Linux, to find the platform architecture, use the operating system command:

```
$ uname -a
```

For 64-bit architecture, it will show x86_64, and x86 for 32-bit architecture.

For MS Windows, use SYSTEMINFO command as shown below:

```
C:\>systeminfo | findstr /i TYPE:
OS Build Type:                Uniprocessor Free
System type:                  X86-based PC
```

Oracle database version

For Oracle source database where the change data capture process is running, the database supplemental logging must be enabled. And, the database should be operating on archive log mode. To find the database version, run the following SQL*Plus query.

```
SQL> SELECT banner
      2 FROM v$version;
```

BANNER

```
-----
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit
Production
PL/SQL Release 12.1.0.1.0 - Production
CORE      12.1.0.1.0    Production
TNS for Linux: Version 12.1.0.1.0 - Production
NLSRTL Version 12.1.0.1.0 - Production
```

```
SQL>
```

MS SQL Server database version

To determine the source MS SQL Server database version, from Microsoft SQL Server Management Studio, execute the query:

```
SELECT @@VERSION
```

The source MS SQL Server database must be operating on Full Recovery model mode. Right mouse on database properties and select Options page as shown on

figure 1-10. However, if the target database is for read-only purpose, it may be operated on Simple model mode to avoid growing the log file unnecessarily when performing long running operation such as initial-load.

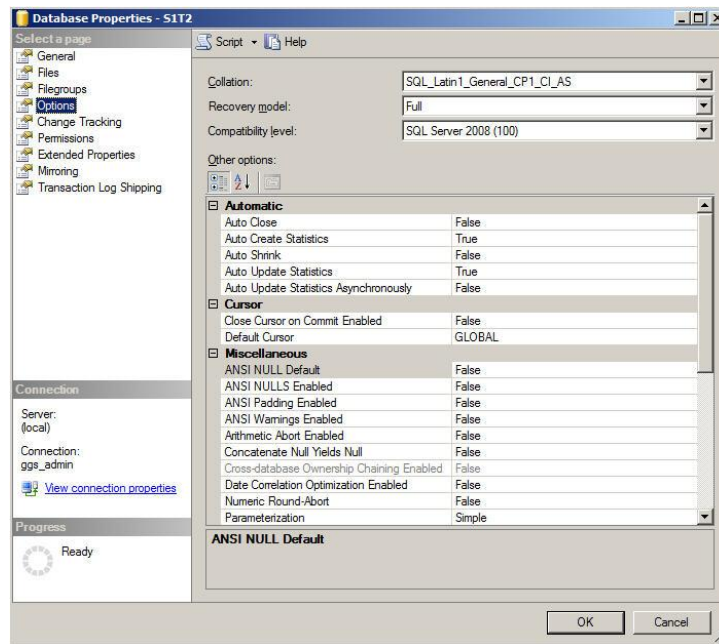


Figure 1-10: MS SQL Server recovery model

The manager process

Each Oracle GoldenGate instance has a dedicated manager process. After successfully installing Oracle GoldenGate, the manager process is the first process configured and started for managing a running Oracle GoldenGate instance. The manager process listen on specific static TCP/IP port and continues running for handling the instances activities. Starting the manager process requires adding the PORT parameter only to mgr.prm file. Refer to chapter 3 to learn more about setting the managers process parameters.

The manager process is responsible for the following activities:

- Response to errors, events and produces reports
- Manages and allocates ports for dynamic processes
- Start Oracle GoldenGate processes
- Start dynamic Oracle GoldenGate processes
- Performs trail files purge management

- The status of the manager process is limited to either of the following status:
 - STOPPED, manually stopped or unable to start
 - RUNNING, started successfully and listening on specific TCP/IP port

The manager process configuration enables automatic start and restart of extract and replicate processes, this provides unattended extract management for enhancing Oracle GoldenGate high availability.

Oracle GoldenGate manager process on Microsoft SQL Server runs as Windows service, the default services name is GGSMGR. To create multiple Windows Oracle GoldenGate instances, use the GLOBALS parameter MGRSERVNAME <service name> for defining unique Oracle GoldenGate Windows service for each instance.

While Oracle GoldenGate instance is running, shutdown the manager process, quickly make the modifications to mgr.prm and restart it. This technique enhances Oracle GoldenGate availability when re-configuring the manager process.

The capture and deliver processes

There are four types of Oracle GoldenGate processes, depending on the task, the topology adapted, extracts and replicates processes are configured and defined.

Online extract

The online extract process perform change data capture (CDC) for committed transactions. The online extract reads the transaction logs from either the online log file or the archive log files. The online redo log files has the advantage of performance for sub-second latency, which cannot be achieved by the archive log file option. The two capture options are mutually exclusive. Figure 1-11 compare the two options. Best practices recommend using online redo log capture with data pump option for fault tolerance and real-time data replication.

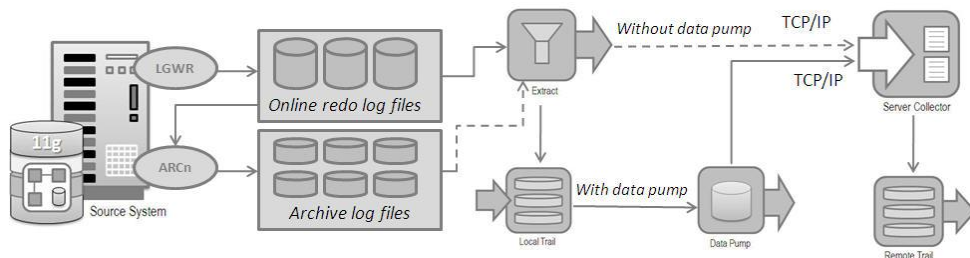


Figure 1-11: Online extract process options

The online extract either writes to local trail files and uses a data pump extract to route data in large block across TCP/IP to a remote dynamic server collector, or across TCP/IP sends the data directly to a remote dynamic process on the target system called data server collector.

The status of the online extract is one of the following:

- STOPPED: The extract process stopped using STOP command
- STARTING: The manager invoking the extract process
- RUNNING: The extract process currently running, reading the transaction log
- ABENDED: The extract process terminated abnormally

The online extract checkpoints it's reads and writes progress to checkpoint file. In the event of failure, the extract process is capable to recapturing from the point of failure forward of the transaction logs.

Initial-load extract

Reads the source database tables for establishing the target database tables. When coordinated with an online extract for change data capture, an initial-load extract enables high availability as it does not lock the source database tables.

An initial-load extract has several options:

- Writes to remote initial-load files
- Directly invoke initial-load replicate process
- Invokes an initial-load replicate process as remote task

Initial-load extract does not maintain checkpoint, in the event of failure the initial-load extract must be restarted after truncating the target tables. Oracle GoldenGate initial-load techniques is suitable for heterogeneous environment, when the source and target database platforms are different. Oracle GoldenGate techniques for initial-load are:

- File to replicate
- Direct load
- File to database utility
- Database utility

Refer to chapter 3 to learn more about initial-load techniques and architecture using GoldenGate and non-GoldenGate methods.

Online replicate

The online replicate process reads the remote trail files and applies transactions to the target database, ensuring the transaction committed order of the source database for data consistency and integrity. The online replicate process expects canonical formatted trail files.

The status of the online replicate process is one of the following:

- **STOPPED:** The extract process stopped using STOP command
- **STARTING:** The manager invoking the extract process
- **RUNNING:** The extract process currently running, reading remote trail files
- **ABENDED:** The extract process terminated abnormally

The online replicate checkpoints its read and write progress using either of two options:

- Concurrently to checkpoint file and database table, known as checkpoint table
- Checkpoint file only

It's highly recommended to use checkpoint file and checkpoint table, which is the default. In the event of failure, the replicate process is capable to restart from the point of failure. By repositioning the relative byte address (RBA), the replicate process is capable of re-applying the remote trail transactions to the remote database.

Special run replicate

Special run replicate is used by the initial-load extract. A special run replicate is started by an initial-load extract and terminated upon completion of the initial-load extract. The special run replicate does not maintain checkpoint, in case of failure the process needs to be restarted.

Remote task replicate

Designed for initial-load. This is used when the initial-load extract directly invokes a remote replicate for performing the initial-load replicate. When invoking a replicate as a remote task, the data collector server is not used.

Note, when using these for initial-load methods on heterogeneous environment such as from Oracle database to MS SQL Server, a source definition file is required to be created at the source system, then transferred to the target system as ASCII file.

Data pump extract

The data pump is an optional extract process, referred to as secondary extract. It reads the local trail files and routes data to the target system. It's highly recommended to use a data pump configurations, the advantage of configuring a data pump extract include:

- In the case of network failure, the primary extract continues writing to the local trail, up on network availability, the data pump process starts sending data to remote system. Without a data pump the primary extract stores change data capture in memory until the network become available. This will cause the primary extract to eventually ABENDED due to system resources unavailability.
- Data movement with intensive data transformation occurring at the source system, the data pump extract offloads the primary extract and leaving it dedicated for change data capture only. TCP/IP communication with the data server collector become a function of the data pump extract only.
- For environment composed of multiple target systems, multiple data pumps read the same local trail and route data to the target systems. When one of the connection is not available, only that specific remote system effected. Without data pump it would require using multiple primary extracts each sending data to specific target system data server collector.
- For security and offloading purposes, implementing an intermediate system architecture is an ideal solution. Two data pump extracts must be configured, on source and intermediate system for routing data. Data transformation occurs at the intermediate system entirely offloading the source system.

The data pump extract not to be confused with Oracle database data pump. This is an Oracle GoldenGate process for reading and routing local trail file to remote systems across TCP/IP network.

The data server collector

The data server collector is a background process, receiving data from either the primary extract (capture) or secondary extract (data pump).

Two types of data server collector are possible, dynamic and statics. Static collector server started manually for the purpose of assigning specific TCP/IP port number to the data collector server, and it can be associated with multiple extract processes. Dynamic collector server is dynamically started, on port allocated by the manager process, and it can be associated with one-only-one extract process.

When using a dynamic mode, starting, stopping and managing the data server collector is fully automatic. The main function is to receive data sent by the extract process, reassemble the data to Oracle GoldenGate universal format (canonical) and write to the remote trail. The data server collector background process is not associated with a checkpoint file, it's either the primary or secondary extract associated with the data collector writing its reads and writes positions to the checkpoint file.

The trails

It's a form of advanced queue structure for managing a temporary storage areas used by Oracle GoldenGate online extract and replicates trail. The default trail files are written in canonical format by the primary extract, and read by the online replicate process. In addition, Oracle GoldenGate supports alternative formats for use by other utilities. Non-default formats:

- `FORMATASCII`, required by file to database initial-load utility
- `FORMATSQL`, generates DML SQL statements
- `FORMATXML`, generates XML files

None of the non-default trail file formats are not supported by the replicate process.

Local trail files

When implementing a data pump option, the local trail files created by the online primary extract of the source system. The local trail files composed of two characters prefix and defined using the `EXTTRAIL` or `EXTFILE` parameters. The local trail file created by appending 6-digit to the trail name. If the local extract trail name is 'sa', then the local trail files takes the ranged-name from sa000000 to sa999999. The default trail file size is 10MB, customized using the `MEGABYTES` parameter of the `ADD EXTTRAIL` or `ADD EXTFILE` commands. An extract must be associated with one local trail only.

Remote trail files

The remote trail files resides on the target system, created by the data collector server. The remote trail files composed of two characters prefix and defined using the `RMTTRAIL` or `RMTFILE` parameters. The remote trail file created by appending 6-digit to the trail name. If the local extract trail name is 'ta', then the remote trail files takes the ranged-name from ta000000 to ta999999. The default trail file size is 10MB, customized using the `MEGABYTES` parameter of the `ADD RMTTRAIL` or `ADD`

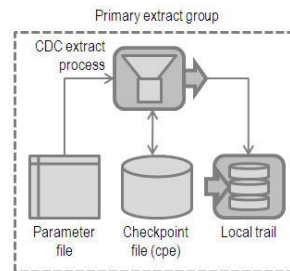
RMTFILE commands. A replicate must be associated with one extract remote trail only.

Processes grouping

The real-time data movement from source to target systems handled by logical groups. The logical grouping allows an effective allocation and management of physical resource associated with each group. Data down-streaming starts at the capture group, optionally through the secondary group and ends at the replicate group. The number of primary, secondary and replicate groups is effected by the database schemas partitioning. The name of the group is limited to eight characters and created by the ADD EXTRACT and ADD REPLICAT commands.

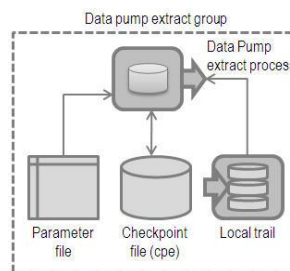
Primary extract group

This is the capture group, it consists of an extract process for performing change data capture (CDC), a parameter file with the name <group name.prm>, a checkpoint file with the name <group name.cpe>, and an output local trail or remote trail files. The group is also associated with its own discard and report files. The primary process captures committed transaction logs, ensuring transaction integrity and data consistency.



Secondary extract group

This is the data pump group, it consist of an extract process for performing data routing, an input parameter file that has the name <group name.prm> and a checkpoint file with the name <group name.cpe> and an input local trail. The group is also associated with its own discard and report files. The data pump uses TCP/IP to communicate with a dynamic dedicated data server collector, which re-assemble and write to the remote trail. This group is optional but recommend for fault-tolerance configuration.



Replicate group

This the change data delivery group, it consists of a replicate process for performing change data delivery, an input parameter file with the name <group name.prm> and a checkpoint table or a checkpoint file with the name <group name.cpr>. The group is also associated with its own discard and report files. The replicate process uses the target database SQL native calls to apply changes from the remote trail, maintaining transactional data integrity data consistency.

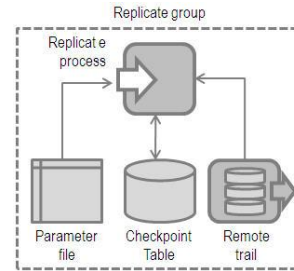


Figure 1-12 depicts the data streaming and the relationship between the three groups, assuming the optional secondary group is used. This diagram shows a serialized data streaming without parallel processing, for details on how Oracle GoldenGate implements parallelism, refer to the section implementing parallel processing.

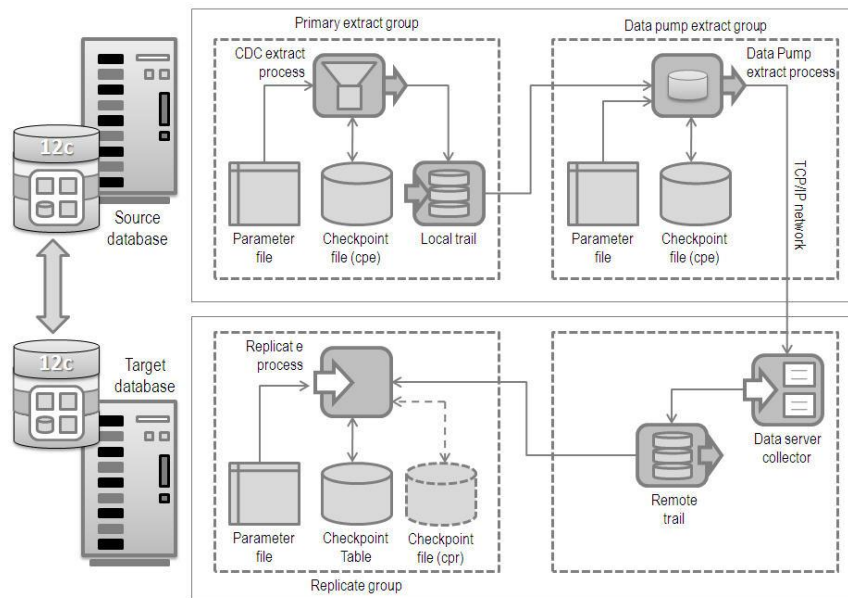


Figure 1-12: Process grouping and workflow

When using intermediate systems, the workflow tends to be loosely coupled, but the concept remains the same. When using integrated capture, the workflow is the same but the change data capture technique is different.

Oracle GoldenGate Architecture explained

The Oracle GoldenGate instance composed of Oracle GoldenGate software location and an associated manager process. Figure 1-13 depicts Oracle GoldenGate architecture on bidirectional environment.

The **manager process**, running on the source and target systems for handling events and starting dynamic extract and replicate processes. Perform trail file purge management for maintaining an optimized disk utilization.

The **primary extract** referred to as the change data capture (CDC) process, reads the transaction logs and write to either a local trail or send data directly to the data sever collector of the remote system. Depending weather a data pump is used or not. The recommended practice is to use a data pump process for sending data to the remote data server collector, keeping the primary extract dedicated for change data capture only.

The **secondary extract** referred to as the data pump process, reads the local trail file and routes data to remote system. The data is received by a dedicated data collector server.

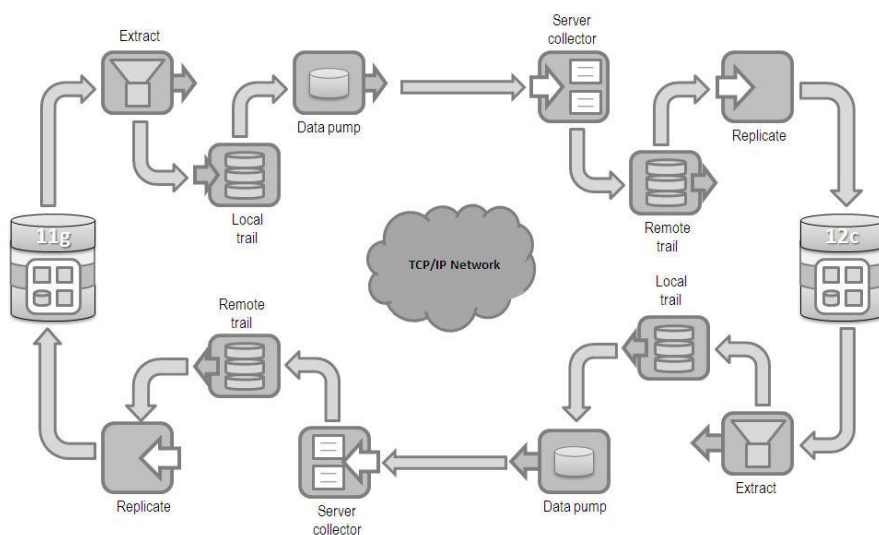


Figure 1-13: GoldenGate processes architecture

The **local trail** is a canonical file format, created by the primary extract, read by the database pump process or other Oracle GoldenGate utilities such as LOGDUMP.

The ***data collector server*** is a background process associated with either a primary extract or a secondary extract. The function of the data collector server is receiving, re-assembling and creating the remote trail on the target system.

The ***remote trail*** is a canonical file format, created by the data collector server. Purging of remote trail files is defined by PURGEAOLDEXTRACTS parameter.

The ***replicate process*** (replicat for short) reads the remote trail file and applying the transactions to the remote database, with data integrity and consistency.

Network topologies and designs

Depending on business data acquisition requirements, the type of Oracle GoldenGate network topology is determined, and it's very important to be the right topology, because changing the topology at later stage requires changing the configuration as it effect data flow and processes. After the topology is selected, the components required to deploy the design are configured, defined, tested and finally deployed. Oracle GoldenGate network topologies listed hereunder followed by description and diagram.

- Unidirectional for query offloading
- Bidirectional for high availability
- Peer-to-peer for load balancing
- Broadcast for data distribution
- Consolidation for data warehouse
- Cascading for scalability

Unidirectional for query offloading

This configuration allows data workflow in one direction, from source to target only. In the source system, Oracle GoldenGate extract groups capture and forward Oracle GoldenGate trails to the remote system. One or more replicate groups apply the transactions to the target database keeping the two databases synchronized. When using unidirectional setup for query offloading, change data capture (CDC) should be limited to the database tables required by the collection of reports, rather than the complete database. Even though offloading to heterogeneous database such as Oracle MySQL may be an option, but it's wise to consider an offloading database engine that offer high-performance reporting capabilities. Oracle database OLAP option, materialized views and query rewrite supercharge database SQL drastically. Figure 1-14 depicts Oracle GoldenGate components involved for unidirectional setup.

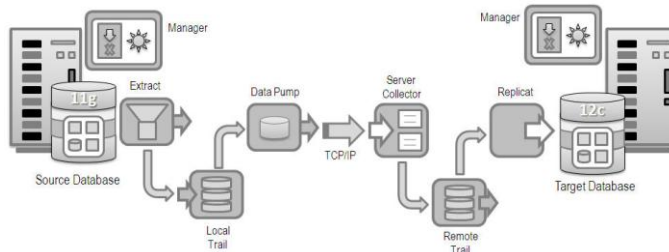


Figure 1-14: Unidirectional Oracle GoldenGate components

Bidirectional network configuration

This configuration enables forward data flow from source to target, and backward data flow from target to source systems. Because Oracle GoldenGate extract and replicate groups are configured on both systems, it's important to handle data conflict as business case scenarios. Oracle GoldenGate supports common data conflict resolutions, for more details regarding data conflict resolutions, refer to chapter 4.

There are two types of bidirectional setup, it's all depending on the business requirements for implementing either option. Figure 1-15 shows the components and data flow for bidirectional configuration. A major advantage of using Oracle GoldenGate for high availability solution is system proximity. Having the source and target systems connected via TCP/IP eliminates the distance restriction. However; network stability and sufficient dedicated bandwidth are key factor to achieve near sub-second latency.

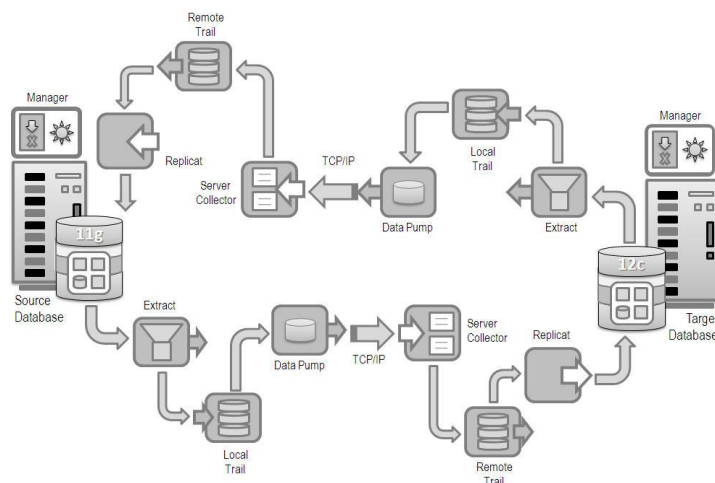


Figure 1-15: Bidirectional configuration

Active-active

Oracle GoldenGate delivers real-time data replication with sub-second latency. Bidirectional active-active setup promotes high availability with customized application scalability, users are able to connect to either system for conducting transactions.

Active-passive

When implementing Oracle GoldenGate for disaster recovery solution, it's configured as active-passive setup. The source and target systems are fully configured to support bidirectional data flow, but at all time, the data flow is active from either directions.

Peer-to-peer configuration

Commonly used for master-to-master data replication. The configuration is used for load balancing and high availability. The two major distinctive features when using Oracle GoldenGate for such design, it can be deployed in heterogeneous environment, for example site A is an Oracle Database 12c, site B is MySQL and site C is Microsoft SQL Server, also supporting replication of subset of the database, this has a positive impact on disk storage and network bandwidth.

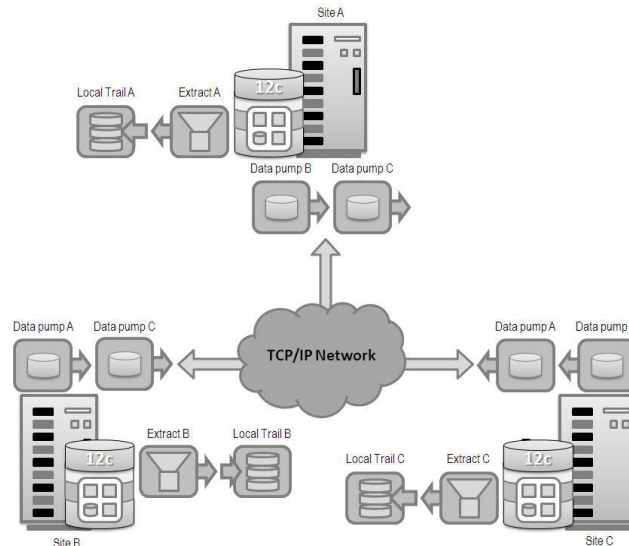


Figure 1-16: Master-to-master replication

Figure 1-16 show master-to-master for three site. Extending this topology by streaming data from a regional master to a local site achieves efficient transaction processing and higher data availability.

The number of sites on peer-to-peer configuration should be limited to 3 sites only. Otherwise, the configuration, data collision handling becomes extremely complex to build, and deploy.

Broadcast configuration

Used for data distribution from the core business applications database to the enterprise data mart databases. This configuration predominately for read-once, deliver-many, after data is captured from the source transaction logs, it can be routed to any number of destinations (targets), regardless of their of their locations and platform types.

Data streaming is unidirectional from source to targets systems, applying data transformations on the data pump processes or on the targets replicate processes customizes the target databases to suite application specific data requirements. Business intelligence applications benefit from real-time data movement and transformation for real-time application support. An important added value from this configuration is security by ensuring physical data separation. The target databases may be small foot-prints of the source, this is very similar to the concept of virtual private database (VPD), but stored remotely. The data is pertaining to subset of related users and/or application for specific region. Figure 1-17 shows the data flow from source to targets.

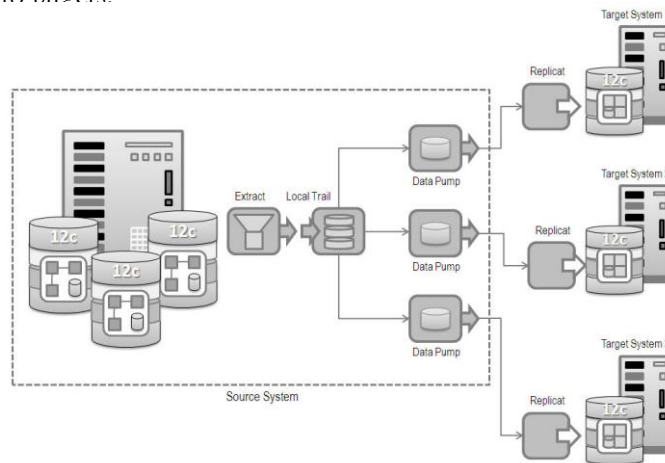


Figure 1-17: Oracle GoldenGate broadcasting topology

Consolidation configuration

Used for the data warehouse, the enterprise data warehouse repository receives data from the enterprise online transactional systems. An E-LT process using Oracle data integrator (ODI) enabling real-time population of the star schemas for reporting and business intelligence. Figure 1-18 shows consolidation of three databases into the data warehouse database.

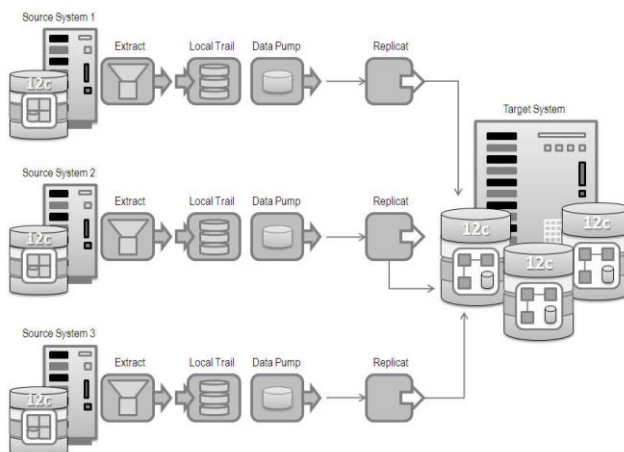


Figure 1-18: Consolidate for data warehouse repository

Cascading configuration

Sites performing intensive data transformation on source and target systems, using an intermediate GoldenGate instance with sufficient processing resources enhances scalability by dedicating the source and target systems for capture and deliver functions only, leaving data transformation to occur within the intermediate system only. The intermediate system can be with or without a database. Figure 1-19 illustrates an intermediate system without a database, only the client software required for installing Oracle GoldenGate software.

The best practices recommends using multiple data pumps, each dedicated to specific target system, network failure of one system is isolated leaving other systems unaffected by the ABENDED data pump process.

In the case of columns mapping or data transformation involving an intermediate system, a source definition file and target definition file must be provided to the intermediate system.

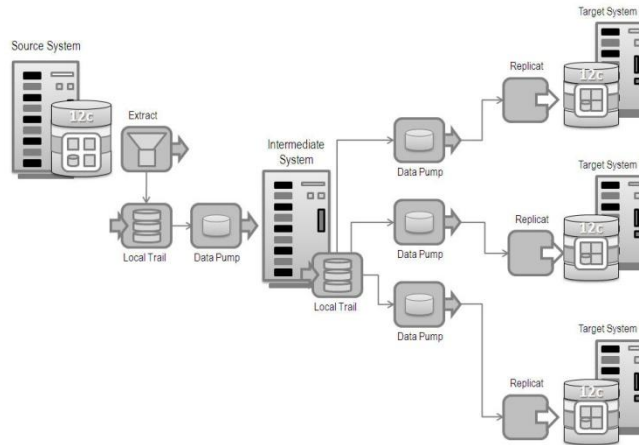


Figure 1-19: Cascade using intermediate system

Oracle GoldenGate command-line interfaces

The Oracle GoldenGate Software command interface, or GGSCI performs administration and management tasks such as adding extracts, adding replicates, trails, login to the database and view reports. On Linux/UNIX operating systems, set the environment variable `LD_LIBRARY_PATH` before starting GGSCI.

```
$export LD_LIBRARY_PATH=$ORACLE_HOME/lib
$cd $GGS_HOME
$./ggsci
```

When using GGSCI for creating and modifying parameter files ensures saving the parameter files on the correct locations. You may also define a non-default preferred text editor. GGSCI is client application tool, it does not require additional software installation, it only uses Oracle database library. Figure 1-20 shows GGSCI on Oracle Enterprise Linux. When using GGSCI for MS SQL Server, run it as Windows administrator user and the manager process runs as a local Windows service.

Examples for using GGSCI interface:

- Manage manager process, performing start, stop and info commands
- Login to the database
- Adding and managing checkpoint table
- Adding and managing extract and replicate processes
- Creating and managing parameter files
- Manages extract and replicate processes
- Generates reports and displaying performance statistics

```
oracle@ggs-source:/u01/app/oracle/ggs - 92x29
$ cd $GGS_HOME
$ pwd
/u01/app/oracle/ggs
$ ./ggsci

Oracle GoldenGate Command Interpreter for Oracle
Version 12.1.2.0.0 17185003 OGGCORE_12.1.2.0.0_PLATFORMS_130924.1316_FBO
Linux, x64, 64bit (optimized), Oracle 11g on Sep 25 2013 00:31:13
Operating system character set identified as UTF-8.

Copyright (C) 1995, 2013, Oracle and/or its affiliates. All rights reserved.

GGSCI (ggs-source) 1> █
```

Figure 1-20: Oracle GoldenGate software command interface (GGSCI)

Another important utility for troubleshooting Oracle GoldenGate instance error messages and performing content-search of trail files is the command-line interface LOGDUMP. Figure 1-21 shows the utility, ready to open a trail file.

The command line to run the utility:

```
$export LD_LIBRARY_PATH=$ORACLE_HOME/lib
$cd $GGS_HOME
$./logdump
```

```
oracle@ggs-source:/u01/app/oracle/ggs - 92x29
$ cd $GGS_HOME
$ pwd
/u01/app/oracle/ggs
$ ./logdump

Oracle GoldenGate Log File Dump Utility for Oracle
Version 12.1.2.0.0 17185003 OGGCORE_12.1.2.0.0_PLATFORMS_130924.1316
Copyright (C) 1995, 2013, Oracle and/or its affiliates. All rights reserved.

Logdump 1 > █
```

Figure 1-21: Oracle GoldenGate log file dump utility

For detail regarding the LOGDUMP utility, refer to chapter 13.

Obey files

The need to run batch job is essential for performing sequence of related tasks, such as running the sequence of commands to deploy an end-to-end Oracle GoldenGate instance. The GGSCI OBEY command reads and execute the commands listed on the text file, known as obey file.

Start by creating a text file, insert one GGSCI command per line and save the file. The example below creates the file `info_all.oby`, then from GGSCI enter the file name as an input to the GGSCI OBEY command. The file name is normally has the file extension `OBY`, but it's not necessary, hence `info_all.oby` and `info_all.txt` are valid file names.

```
-- File: /home/oracle/oby/info_all.oby
--
start mgr
info all
start extract ES1E201
start extract EPMP01
info all
```

```
GGSCI (ggs-source) 7> OBEY /home/oracle/oby/info_all.oby
```

The GGSCI OBEY command allows nesting of obey files for invoking another obey file from within an obey file. To enable nesting, insert the parameter `ALLOWNESTED` at the beginning of the file before invoking and obey file as illustrated next.

```
-- File: /home/oracle/oby/info_all.oby
--       /home/oracle/oby/add_tran.oby
--
ALLOWNESTED
START mgr
INFO ALL
START EXTRACT es1e201
START EXTRACT epmp01
INFO ALL
OBEY /home/oracle/oby/add_tran.oby
```

```
GGSCI (ggs-source) 18> OBEY /home/oracle/oby/info_all.oby
```

```
GGSCI (ggs-source) 19> ALLOWNESTED
```

```
Nested OBEY scripts allowed.
```

```
GGSCI (ggs-source) 20> START mgr
```

```
Manager started.
```

```
GGSCI (ggs-source) 21> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	STOPPED	EPMP01	00:00:00	00:00:29
EXTRACT	STOPPED	ES1E201	00:00:00	00:00:28

```
GGSCI (ggs-source) 22> START EXTRACT es1e201
```

```
Sending START request to MANAGER ...  
EXTRACT ES1E201 starting
```

```
GGSCI (ggs-source) 23> START EXTRACT epmp01
```

```
Sending START request to MANAGER ...  
EXTRACT EPMP01 starting
```

```
GGSCI (ggs-source) 24> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	RUNNING	EPMP01	00:00:00	00:00:32
EXTRACT	RUNNING	ES1E201	00:00:01	00:00:00

```
GGSCI (ggs-source) 25> OBEY /home/oracle/oby/add_tran.oby
```

```
**** Halting script [/home/oracle/oby/info_all.oby], starting  
script [/home/oracle/oby/add_tran.oby]...
```

```
GGSCI (ggs-source) 26> dblogin userid ggs_admin, password oracle
```

```
Successfully logged into database.
```

```
GGSCI (ggs-source) 27> add trandata osm$repapi.customers
```

```
Logging of supplemental redo data enabled for table  
OSM$REPAPI.CUSTOMERS.  
TRANDATA for scheduling columns has been added on table  
'OSM$REPAPI.CUSTOMERS'.  
GGSCI (ggs-source) 28>
```

```
GGSCI (ggs-source) 28> **** Terminating script
[/home/oracle/oby/add_tran.oby], resuming script
[/home/oracle/oby/info_all.oby]...
```

```
GGSCI (ggs-source) 28>
```

However; for executing an SQL*Plus script and operating system commands, use GGSCI SHELL command within an obey file followed by the shell script file. For example, in certain situation, it may be needed to restart the database from within Oracle GoldenGate, which can be developed as follows:

- Create an obey file, db_shut.oby, that has the lines:

```
SHELL /home/oracle/oby/db_shut.sh
```

- Prepare the database shutdown script and save it as db_shut.sh. Ensure the file has execute permission.

```
#!/bin/bash
export ORACLE_SID=S1E2
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1
sqlplus /nolog <<EOF
conn / as sysdba
startup force
exit
EOF
```

- Invoke the obey file from GGSCI command-line.

```
GGSCI (ggs-source) 1> OBEY /home/oracle/oby/db_shut.oby
```

When using GGSCI SHELL command to execute an operating system shell commands, these commands are executed with the access rights of the Oracle GoldenGate software owner, applying access rights of the database privileges and operating system security.

Parameters files

There are two types of parameters, global parameters type apply to all Oracle GoldenGate instance processes, and runtime parameters type apply to specific process. When the same parameter specified as global and runtime, the runtime parameter takes precedence and override the global parameter. Parameter files are ASCII format, with one line per parameter or append the character ‘&’ to create multi-line parameter. The following syntax for the PURGEOLDEXTRACTS parameter produces the same results.

```
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPHOURS 1

PURGEOLDEXTRACTS ./dirdat/*, &
USECHECKPOINTS, &
MINKEEPHOURS 1
```

The global parameters are defined on the file GLOBALS (upper case without file extension) located at Oracle GoldenGate root directory. Global parameters are relatively static and rarely modified. Table 1-3 list common global parameters.

Parameter	Description
CHECKPOINTTABLE	Specifies instance-wide checkpoint table
ENABLEMONITORING	Required for enabling Oracle GoldenGate monitor
MGRSERVNAME	The name of the manager process when installed as local Windows service
CHARSET	The character set to when reading the parameter files, overriding the default operating system
GGSCHEMA	The name of the schema for supporting DDL replication between two Oracle databases
SYSLOG	Used for filtering the messages types recorded in ggserr.log file to override the default ALL.

Table 1-3: Common global parameters

When changing global parameters on GLOBALS file from within GGSCI, it's required to exit and restart GGSCI utility for activating the newly added parameters.

Runtime parameters are specific to an extract or replicate process allowing more control for process runtime. The default location for run-time parameter files is ./dirprm/ directory of Oracle GoldenGate location. Using text editor such as 'vi' on Linux/UNIX, notepad on Windows or GGSCI to create a new or edit an existing parameter file, the group name specified at GGSCI command-line, which becomes the parameter file name, must matches the name followed the EXTRACT parameter. The process name, group name and parameter file name are not case sensitive.

The following listing is the parameter file for extract ES1E201, created using GGSCI command-line.

```
GGSCI (ggs-source) 2> EDIT PARAMS es1e201

EXTRACT ES1E201
SETENV (ORACLE_SID=S1E2)
SETENV (ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD oracle
```

```
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

Configure the instance manager process

A GoldenGate instance can have one-and-only manager process. Which remains running to monitor activities and start dynamic processes. There are few important facts about the manager process.

- The only required parameter is PORT, which specifies a static TCP/IP listening port number.
- There can be one-and-only-one manager process per Oracle GoldenGate instance.
- The manager process status is either RUNNING or STOPPED.
- The manager process parameter file name and location must be ./dirprm/mgr.prm only.
- The manager process supports up to 250 dynamic ports
- Though the manager should remain running at all times, it may be stopped, change the parameter and restart without affecting currently running processes.

The following is an example of common manager process parameters. These parameters' values should be carefully determined as they are reflected on how Oracle GoldenGate instance manager process handles events.

```
-- mgr.prm
--
PORT 7811
DYNAMICPORTLIST 9001-9250
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPHOURS 1
AUTORESTART ER *, RETRIES 10, WAITMINUTES 5
LAGREPORThOURS 2
LAGINFOMINUTES 5
LAGCRITICALMINUTES 10
```

This manager process is listening on TCP/IP port 7811. It searches and allocates from the range of ports 9001 to 9250 for dynamically starting processes.

The parameter PURGEOLDEXTRACTS allows the manager to perform trail files management. The option USECHECKPOINTS to delete trail files after being transferred to the target system and applied. The option MINKEEPHOURS 1 instructs to delete the trail file until there has been no activity for one hour.

The manager will automatically start any extracts and replicates processes at startup and will attempt to restart any extracts and replicates with status **ABENDED** after waiting for five minutes, but only up to ten attempts.

The parameter **LAGREPORTHOURS 2** instructs the manager to report lag statistics every two hours for those processes that have five and ten minutes of latency using the parameters **LAGINFOMINUTES 5** and **LAGCRITICALMINUTES 10**. The message recorded as informational for lags of five minutes and critical for lag greater than ten minutes.

To find the basic details about the currently running manager, issue the GGSCI command:

```
GGSCI (ggs-source) 8> INFO mgr
```

```
Manager is running (IP port ggs-source.7811, Process ID 21649).
```

To view the manager process parameters, issue the GGSCI command:

```
GGSCI (ggs-source) 9> VIEW PARAMS mgr
```

```
PORT 7811
DYNAMICPORTLIST 9001-9250
PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPHOURS 1
AUTORESTART ER *, RETRIES 10, WAITMINUTES 5
LAGREPORTHOURS 2
LAGINFOMINUTES 5
LAGCRITICALMINUTES 10
```

To issue stop and restart the manager process, enter the GGSCI commands:

```
GGSCI (ggs-source) 12> STOP mgr!
```

```
Sending STOP request to MANAGER ...
Request processed.
Manager stopped.
```

```
GGSCI (ggs-source) 13> START mgr
Manager started.
```

```
GGSCI (ggs-source) 14>
```

Even though the manager process may be stopped without effecting the currently running extracts and replicates processes, it should remain running at all time to handle runtime events and activities. A common case is when very quickly changing a

specific parameter value in the manager file, stopping and restarting the manager process immediately.

Refer to chapter 3 for more details regarding the parameters description.

Configure change data capture

The change data capture is a function of the primary extract, during the capture operation the transaction and data remain in memory until the transaction is committed or rolled back. Committed transactions are written to the local trail in case a secondary extract (data pump) is configured or to the remote trail, otherwise. Best practices states configuring a data pump provides resiliency against network failures.

The configuration below instructs the primary extract to perform change data capture for the database tables CUSTOMERS and POLICIES.

```
EXTRACT es1e201
SETENV (ORACLE_SID=S1E2)
SETENV
(ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD oracle
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

The primary extract group name must matches the parameter file name es1e201.prm. The SETENV parameter access environment variable is required for operating the primary extract. The parameter USERID enables the primary extract to login to the database S1E2. The parameter EXTTRAIL instructs the primary extract to write to the local trail ./dirdat/sa. The TABLE parameter lists the tables for change data capture.

Oracle GoldenGate adapts its own universal format for exchanging data across heterogeneous systems. The parameter EXTTRAIL specifies the location and the name of the local trail files. The name is a two character prefix appended with 6 digit starting 000000 and ending with 999999, then recycling begin at 000000.

Creating the primary extract

From GGSCI command-line use the ADD EXTRACT command to define the primary extract. The parameter file may not exist when adding the extract, but

required to start it. The ADD EXTRACT syntax below is used for defining the primary and secondary extracts.

```
ADD EXTRACT <group name>
{, SOURCEISTABLE |
, TRANLOG [<bsds name>] |
, INTEGRATED TRANLOG |
, VAM |
, EXTFILESOURCE <file name> |
, EXTTRAILSOURCE <trail name> |
, VAMTRAILSOURCE <VAM trail name>}
{, BEGIN {NOW | yyyy-mm-dd [:hh:mi:[ss[.cccccc]]]} |
, EXTSEQNO <seqno>, EXTRBA <relative byte address> |
, LOGNUM <log number>, LOGPOS <byte offset> |
, EOF |
, LSN <value> |
, EXTRBA <relative byte address> |
, EOF | LSN <value> |
, PAGE <data page>, ROW <row>}
[, THREADS <n>]
[, PASSIVE]
[, PARAMS <parameter file>]
[, REPORT <report file>]
[, DESC "<description>"]
```

The ADD EXTRACT command has three sections as highlighted above: The extract type, the capture begin timestamp or absolute location and the optional control parameters. For defining the primary extract ES1E201, we will use the basic parameters.

```
GGSCI (ggs-source) 25> ADD EXTRACT es1e201, TRANLOG, BEGIN  
now, THREADS 1  
EXTRACT added.
```

The above ADD EXTRACT command has successfully defined the extract group ES1E201. This extract performs change data capture from the transaction logs (online redo log files) indicated by the option TRANLOG. The timestamp to begin change data capture is the time of up on successful completion of the command indicated by option BEGIN NOW. The option THREADS 1 indicates the extract captures from a single thread on non-RAC Oracle database instance.

Using a secondary extract requires creating a local trail. The local trail ./dirdat/sa is associated with the primary extract ES1E201.

```
GGSCI (ggs-source) 26> ADD EXTTRAIL ./dirdat/sa, EXTRACT
es1e201, MEGABYTES 32
EXTTRAIL added.
```

After change data capture by the primary extract process written to the local trail, then the role of the secondary extract process (data pump) begins by routing data to one or more target systems. The following refers to the data pump process configuration.

```
EXTRACT epmp01
USERID ggs_admin@s1e2, PASSWORD oracle
PASSTHRU
RMTHOST ggs-target, MGRPORT 7812
RMTTRAIL ./dirdat/ta
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

The data pump process epmp01 reads the local trail files. The parameter PASSTHRU indicate the source and target database tables structure are identical. NOPASSTHRU parameter is used when using the @RANGE function to support parallel processing or COLMAP parameter implemented for non-identical tables.

The parameter RMTHOST specifies the target system, the MGRPORT is the target system manager process.

The parameter RMTTRAIL specifies the location and name of remote trail on the target system.

Creating the secondary extract

Define the secondary group using ADD EXTRACT GGSCI command. The secondary extract reads the local trail and routes data across TCP/IP to remote data collector server.

```
GGSCI (ggs-source) 27> ADD EXTRACT epmp01, EXTTRAILSOURCE
./dirdat/sa
EXTRACT added.
```

The above ADD EXTRACT command define the secondary extract EPMP01, the extract reads the local trail ./dirdat/sa indicated by the option EXTTRAILSOURCE. The remote trail ./dirdat/ta associated the secondary extract defined using the ADD RMTTRAIL command.

```
GGSCI (ggs-source) 28> ADD RMTTRAIL ./dirdat/ta, EXTRACT
epmp01, MEGABYTES 32
```

RMTTRAIL added.

Use the command `START EXTRACT *` for starting all non-started extracts. Then find the current status of the extracts by issuing `INFO ALL` command. This `INFO ALL` command shows the process status, and more importantly lagging and checkpoint details.

```
GGSCI (ggs-source) 34> START EXTRACT *
```

```
Sending START request to MANAGER ...  
EXTRACT EPMP01 starting
```

```
Sending START request to MANAGER ...  
EXTRACT ES1E201 starting
```

```
GGSCI (ggs-source) 35> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	RUNNING	EPMP01	00:00:00	00:00:13
EXTRACT	RUNNING	ES1E201	00:00:01	00:00:00

```
GGSCI (ggs-source) 36>
```

Refer to chapter 3 for more details regarding extracts parameters.

Configure change data delivery

The third and last step of the configuration is the change data delivery, the replicate process. The replicate process uses the target database native SQL calls to apply the transactions found on the remote trail. Reapplying the remote trail transactions is subject to the database constraint definition for primary keys and referential integrity.

The following is the change data delivery group parameter file.

```
REPLICAT rs1e201  
SETENV (ORACLE_SID=T1C1)  
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)  
USERID ggs_admin@t2c1, PASSWORD oracle  
ASSUMETARGETDEFS  
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE  
MAP osm$repapi.customers, TARGET osm$repapi.customers;  
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

The replicate process RS1E201 connects to Oracle Database 12c, it references the list of mapped tables for applying transactions to the target database using its associated remote trail files. The ASSUMETARGETDEFS parameter is for homogenous environment when using identical source and target database tables structure. The MAP and the corresponding TARGET parameters are used to map the source with target table.

Creating change data delivery process

The change data delivery is the replicate process reading from remote trail and applying transactions to the remote database using database native SQL calls. The ADD REPLICAT syntax is used to define change data delivery group.

```
ADD REPLICAT <group name>
{, SPECIALRUN |
, EXTFILE <file name> |
, EXTTRAIL <trail name>}
[, BEGIN {NOW | yyyy-mm-dd:hh:mm[:ss[.cccccc]]} |
, EXTSEQNO <seqno>, EXTRBA <rba>]
[, CHECKPOINTTABLE <owner.table> | NODBCHECKPOINT]
[, PARAMS <parameter file>]
[, REPORT <report file>]
[, DESC "<description>"]
```

The ADD REPLICAT command has three sessions: The replicate process type and data source, the apply begin timestamp and the optional control parameters.

Use the basic parameter to create the replicate group RS1E201, specifying the data source and the database checkpoint table name. The replicate process requires checkpoint database table, by default. Using the default option requires creating the checkpoint table before defining the replicate process RS1E201.

```
GGSCI (ggs-target) 4> ADD CHECKPOINTTABLE  
ggs_admin.ggchkpttab
```

Successfully created checkpoint table ggs_admin.ggchkpttab.

```
GGSCI (ggs-target) 16> ADD REPLICAT rs1e201, EXTTRAIL  
./dirdat/ta, CHECKPOINTTABLE ggs_admin.ggchkpttab  
REPLICAT added.
```

Use the command START REPLICAT to start the replicate process, then issue INFO ALL to confirm status of the replicate process.

```
GGSCI (ggs-target) 19> START REPLICAT *
```

```
Sending START request to MANAGER ...  
REPLICAT RS1E201 starting
```

```
GGSCI (ggs-target) 20> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RS1E201	00:00:00	00:00:53

```
GGSCI (ggs-target) 21>
```

Refer to chapter 3 for more details regarding parameter description and database checkpoint table options.

Using macros

Macros enable an easier and more efficient method for building Oracle GoldenGate instance parameter files. A macro defines a group of related parameters so that they can be referenced from within the parameter files, simplifying the configuration of large parameter files with repetitive parameters, commands and functions by reusing the group of parameter defined by the macro.

The definition of macros either *local* within the same process parameter file and invoked from within parameter file only, or *global* to Oracle GoldenGate instance by defining macros library and invoking the macros from any extract or replicate processes parameter files.

Prior to invoking macros, they must be defined. The definition of a macro similar to sub-programs, where a macro has a name beginning with # character, optionally followed by zero or more parameters. The macro body placed between the BEGIN and END statements. The syntax for defining macros follows.

```
MACRO #<macro name>  
PARAMS (#<parameter1>, #<parameter2>, #<parameter>)  
BEGIN  
    <Macro body>  
END;
```

After defining a macro, it can be used on the parameter files for:

- Manager process
- Primary and secondary extracts
- Replicate process
- GLOBALS parameter file

The best approach to work with macros is by creating macro libraries. Let's assume the parameters LOGIN and PASSTHRU are included for the majority of parameter files. Defining and invoking a macro replaces explicitly adding the those parameters to the parameter files. The parameter INCLUDE at the top of the parameter file enables invoking any of the macros defined within the macro library file.

- Start by defining the macro named #S1E2LOGIN inside the macro library file

```
$ vi /u01/app/oracle/ggs/dirprm/macrosqe2.mac
```

```
MACRO #S1E2LOGIN
BEGIN
    USERID ggs_admin@S1E2, PASSWORD oracle
    PASSTHRU
END;
```

- Invoke the macro from within the secondary extract EPMP01

```
INCLUDE /u01/app/oracle/ggs/dirprm/macrosqe2.mac
EXTRACT epmp01
#S1E2LOGIN()
RMTHOST ggs-target, MGRPORT 7812
RMTTRAIL ./dirdat/ta
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

The modular architecture

Regardless of the design adapted to meet the business requirements for data acquisition, the level of database schema complexity, the data flow option, the database heterogeneity; The modular building blocks of Oracle GoldenGate is capable to produce a dynamic architecture for achieving real-time data movement with sub-second latency. The capture-and-apply processes operate as loosely or tightly coupled, loosely coupled in the sense the capture processes may be cascaded across multiple systems, where the primary extract process (capture) running on the source database system reading the transaction logs, and the secondary extract process (data pump) running on an intermediate system physically networked with the source system using TCP/IP. These capture-apply processes runs outside of the database management

system, hence Oracle GoldenGate delivers safe, high performance and scalable platform with minimum impact on existing system for supporting heterogeneous data movement.

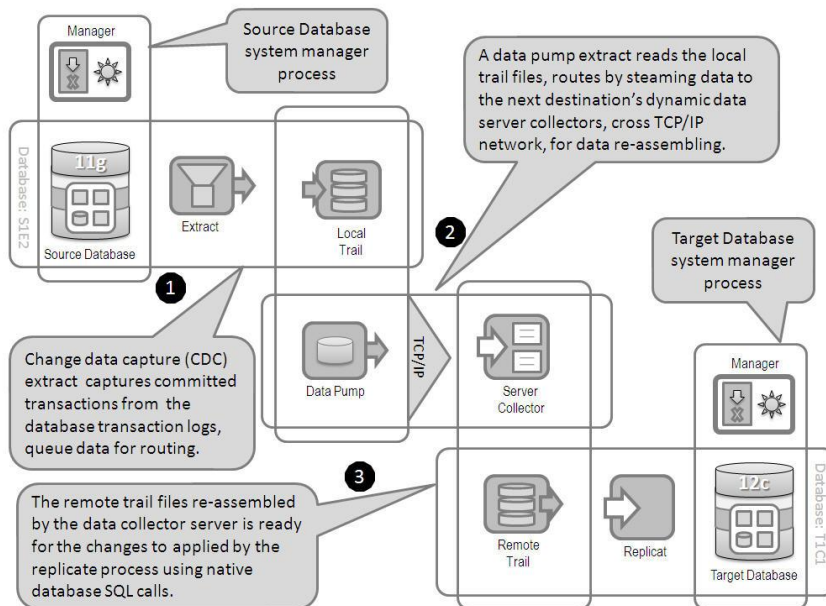
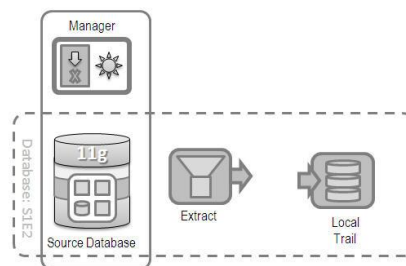


Figure 1-22: Oracle GoldenGate modular architecture

Figure 1-22 illustrates the building block for the three modular blocks:

- Change data capture phase using primary extract group
- Routing phase using secondary extract group
- Apply phase using change data delivery group

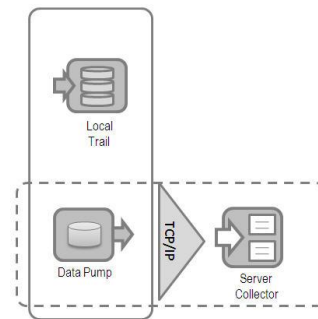
The capture phase, it involves one or more primary extract capturing *only committed* transactions as they become available on the transaction logs. This is occurring outside of the database, hence minimum impact sustained by the source database. For achieving a fault-tolerance non-interruptible data movement, the primary extract stages captured data in the form of advanced queues composed of local trail files ready for routing to one or more



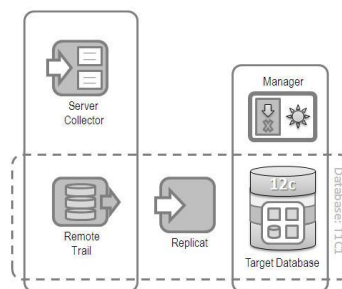
destinations (targets). The target systems could be the final destinations (sources) or an intermediate systems for achieving high-performance broadcasting network topology or intensive data transformation.

The source system manager process normally configured to manage local trail files purge task for sustaining an optimized disk space on the production system. It's also configurable to implement extract processes auto-start during system startup and restart for abnormally terminated extract processes. Trail files purge and extracts auto-start features are implemented for minimizing data movement discontinuity on sites requiring uninterrupted data processing.

The routing phase, it involves one or more secondary extracts called data pumps for routing transactions, *not trail files*, to one or more destinations. Having a data pump within the source system is optional, but very highly recommend to support fault-tolerance as explained on the section – Capture-route options. The receiving component is the data server collector for re-assembling the data and managing the remote queue. The data server collector is a dynamic server and associated with a data pump. The communication from the data pump extract to the data server collector occurs across TCP/IP network allowing the design for loosely coupled system where the source and target systems are geographically distributed.



The third, or last phase involves one or more change data apply, replicate processes know by Replicat. Replicate processes running on the target system use the database interface (SQL calls) for applying the transactions while maintaining transactions ordering for data integrity. The input of the replicate processes is the remote trail files created by the data collector server. High-performance apply is achieved using Oracle GoldenGate flexible parallelism architecture allowing multiple replicate processes to read from a single trail composed of trail files.



The manager process on the target system normally configured to manage remote trail files purge task for sustaining an optimized disk space on the target system. It's also configurable to implement replicate processes auto-start during system startup and restart for abnormally terminated replicate processes.

The flexibility of designing an operational Oracle GoldenGate instance is depicted logically on figure 1-23 followed by the interpretation.

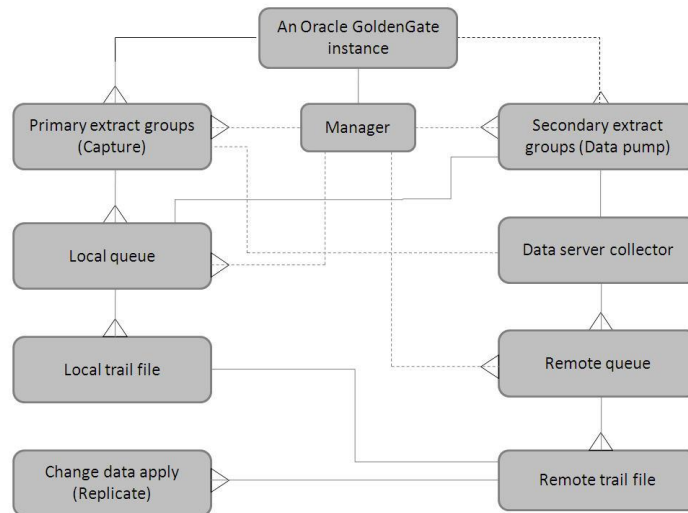


Figure 1-23: GoldenGate components relationship

An Oracle GoldenGate instance must have one-and-only-one manager. An *operational* Oracle GoldenGate instance must be composed of one-or-more primary extract groups (capture), but may have one-or-more secondary extract groups (data pump).

Each primary extract must create one-and-only-one trail, and each trail must be composed of one-or-more local trail files. Each secondary extract (data pump) is associated with one-and-only-one data server collector and each data server collector creates one-and-only-one remote trail. Each remote trail composed of one-or-more remote trail files.

An Oracle GoldenGate instance must be composed of one-or-more replicate processes, and each trail file is applied using the target database SQL native interface.

The manager process may manage one-or-more primary extracts, one-or-more secondary extracts and one-or-more trails.

Capture-route options

There are two options available for deploying Oracle GoldenGate as visually compared in figure 1-24. The upper section shows a primary extract (capture) with a

data pump extract refers to as option-A and the lower section shows a primary extract (capture) without a data pump extract refers to as option-B. Best practices highly recommends implementing the primary extract with a data pump extract, option-A.

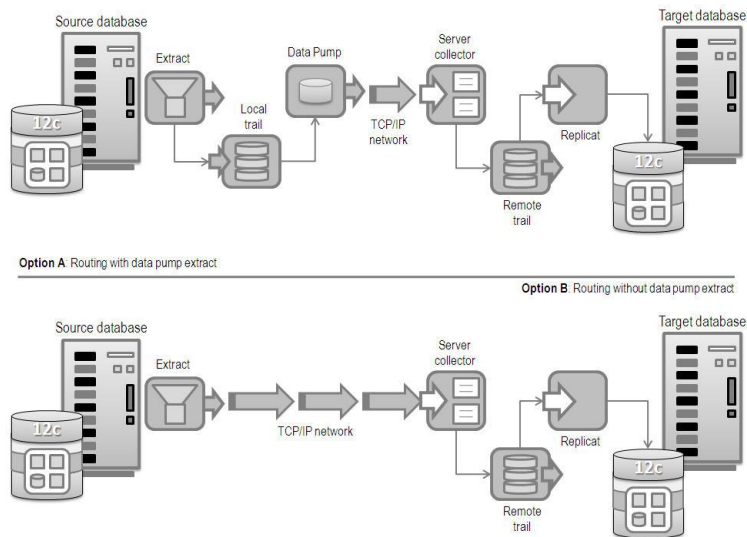


Figure 1-24: End-to-end configuration options

In the event of TCP/IP network disconnection from the source to target systems, option-A supports fault-tolerance because the primary extract continues to deliver change data capture (CDC) by writing to the local trail, only the data pump extract will be terminated temporary. Once TCP/IP network restored, the data pump is restarted and data continue sending data to the data server collector. Under the same circumstances, option-B will cause the primary extract process to terminate due to system resources unavailability as a result of unable to send data to the remote data server collector. Depending on the downtime timeframe, the transaction logs may not be available after the network is restored and the extract started. This scenario requires manual changes for reading the transaction logs from archived log files.

The storage overhead of writing to local trail files is minimized by the purge task automatically performed by the manager process and the parameters used to configure the purge. In most cases, under normally working extract-route-replicate for option-A the local trail files disk space is negligible and should not cause performance degradation. Option-B is selected in case where the source and target Oracle GoldenGate instances are running on the same host, moving data between two heterogeneous or homogenous databases.

Commit sequence number

The commit sequence number, CSN identifies a particular point in time of the source transaction during the capture. It's constructed from the database, for Oracle database it's the same as the system change number (SCN), for MS SQL Server it's composed of three parts: the virtual log file number, the segment number within the virtual file and the entry number. Oracle GoldenGate uses CSN during apply to identify the point in time at which the transaction is committed for maintaining transaction consistency and data integrity.

Checkpoints

Primary extract, data pump extract and replicate processes record their reads and writes positions along the data flow. Successfully created extract and replicate groups are associated with a checkpoint file. The checkpoint file name for an extract group is <group name.cpe>, the checkpoint file name for a replicate group is <group name.cpr>. For primary extracts performing change data capture save their writes and reads positions to a checkpoint file to prevent redundant extraction from the transaction logs. For data pump extracts performing routing to remote destination across TCP/IP save their reads and writes positions to prevent redundant extraction of the local trail. For replicates processes performing change data apply save their reads and writes positions to a checkpoint table or checkpoint file to prevent redundant extraction from the remote tail files and reapply to the target database.

Oracle GoldenGate checkpoint ensures fault tolerance in case of failure. The recovery process is similar to Oracle database automatic instance recovery.

For primary and data pump extracts checkpoint file is the only option. For replicates, there are two options; checkpoint file or checkpoint database table and checkpoint file combined, the default is the combined option. The name of the table either specified on GLOBALS file for all replicate groups using the parameter CHECKPOINTTABLE followed by the schema and the table name, or per replicate group specified when adding the replicate process using GGSCI command ADD REPLICATE. To override the default and use a checkpoint file for the replicate group, use the option NODBCHECKPOINT when adding the replicate process through GGSCI command ADD REPLICATE.

Intermediate systems

For achieving an enterprise scalability in term of data transformation and routing capabilities, the use of Oracle GoldenGate instance as an intermediate system

receiving very active downstream of trail files from multiple source systems, then transforming and routing trail files to multiple remote systems. An intermediate system is configurable as:

- Non-database intermediate system
- Database intermediate system

Figure 1-25 depicts intermediate system, it's composed of an Oracle GoldenGate instance configured with data pump processes for transforming and routing data to target systems.

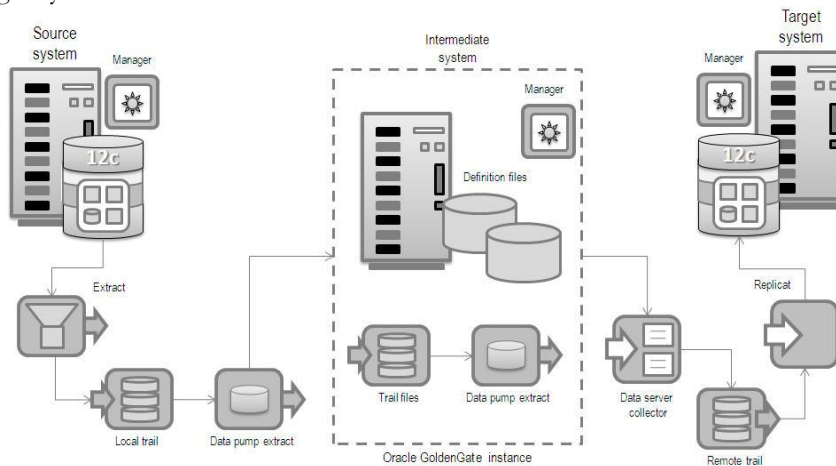


Figure 1-25: Intermediate system for scalability

In addition to achieving best scalability using an intermediate system, it also support enhanced security. Which is true in the case the target system receiving data from non-trusted source systems. With intermediate system in place, prior to routing the data to the target database, the data is verified, transformed then sent over secure channel to the target database. The intermediate system uses multiple data pump extracts to route data to multiple targets, and has data definition files for each target requiring data transformation before routing to the target database, the advantage of performing transformation within the intermediate system is to offload the target database from the overhead associated with intensive data transformation and filtering.

Integrated capture

The purpose of the integrated capture is to offload the transaction logs processing by Oracle GoldenGate from the source database (production) to a downstream system.

Integrated capture is a specialized form of intermediate system continuously receiving the source database REDO as standby log files, where the integrated capture process on the downstream system performs change database capture (CDC) and forwarding the transaction data to the next downstream process such data pump, which in turn routes data to the target system. Figure 1-26 shows the architecture of the integrated capture environment.

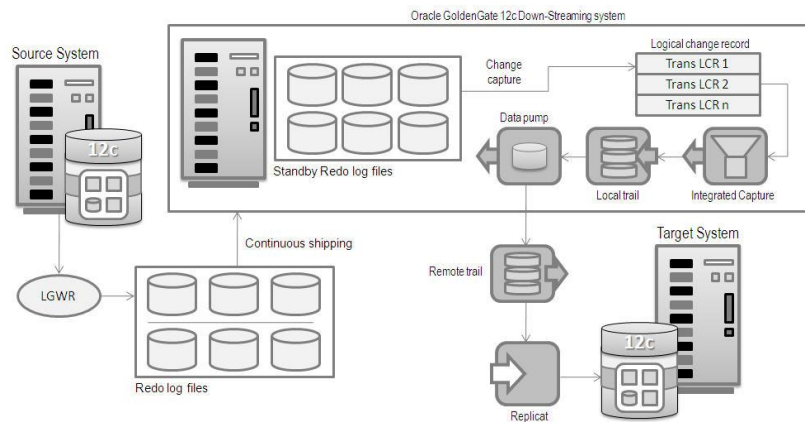


Figure 1-26: Integrated capture architecture

There are three options of integrated capture architecture, available with Oracle Database version 10.2 or higher.

- Option 1: Real-time same system integrated capture
- Option 2: Real-time separate system integrated capture
- Option 3: No-real-time separate system integrated capture

Real-time same system integrated capture requires installing Oracle GoldenGate 12c on source system, the integrated capture delivers remote trails from the source database REDO logs files.

Real-time separate system integrated capture, Oracle GoldenGate is not installed on the source system, but on a downstream system continuously receiving the source database REDO log files, where the integrated capture process delivers the remote trails on real-time.

No-real-time separate system integrated capture, Oracle GoldenGate is not installed on the source system, but on a downstream system continuously receiving the source database REDO log files, but Oracle GoldenGate integrated is operating on archive log only (ALO) mode readings the standby archived log file.

Refer to chapter 9 to learn more about Oracle GoldenGate integrated capture.

Columns mapping

It's the norm having the structure of source database table differs from the target database table requiring handling using columns mapping techniques. This is applicable to tables with different column names, missing column names and different columns ordering. Use the TABLE or MAP statement COLMAP option to develop the mapping for each pair of columns.

The key elements of columns mapping are:

- Source definition file, generated using DEFGEN Oracle GoldenGate utility
- Mapping clause: COLMAP with USEDEFAULTS or COLMATCH

Let's illustrate columns mapping to handle different column names on the CUSTOMERS table for the two columns CUST_ADDRESS and CUST_ADDR.

The source database CUSTOMERS table

```
S1E2>DESC customers
Name                               Null?    Type
-----
CUST_NO                           NOT NULL NUMBER
CUST_NAME                         NOT NULL VARCHAR2 (128)
CUST_EMAIL                       NOT NULL VARCHAR2 (128)
CUST_MOBILE                       NUMBER
CUST_ADDRESS                    VARCHAR2 (128)
```

The target database CUSTOMERS table

```
T1C1>desc customers
Name                               Null?    Type
-----
CUST_NO                           NOT NULL NUMBER
CUST_NAME                         NOT NULL VARCHAR2 (128)
CUST_EMAIL                       NOT NULL VARCHAR2 (128)
CUST_MOBILE                       NOT NULL VARCHAR2 (16)
CUST_ADDR                      VARCHAR2 (128)
```

Definition file utility

For heterogeneous data replication and replicating dissimilar tables on homogenous environment, a definition file created using Oracle GoldenGate DEFGEN utility is required. The utility generates the a source file describing the structure of the source to enable Oracle GoldenGate handles the internal data type mapping.

The steps to use the DEFGEN utility for generating a source definition file and apply it to the replicate process parameter file follows.

- Prepare the parameter file defgen.prm. Save the parameter file on /dirprm directory.

```
DEFSFILE ./dirdef/oracle.def, PURGE
USERID ggs_admin, PASSWORD oracle
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

- Run the utility using the parameter file. The utility creates the definition file specified by the parameter DEFSFILE.

```
$cd $GGS_HOME
$ ./defgen PARAMFILE ./dirprm/defgen.prm REPORTFILE
./dirrpt/defgen.rpt
```

- Using ASCII mode, transfer the source definition file oracle.def to the target system into the directory ./dirdef
- Modify the replicate parameter file to include the parameter SOURCEDEFS, and remove the parameter ASSUMETARGETDEFS.

```
REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@t1c1, PASSWORD oracle
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS, cust_addr = cust_address);
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

When using the DEFGEN utility to produce the source definition file, the character set of the source and target systems presumably matching. However; to guarantee the

definition file generated compatible with the target system character set, specify the CHARSET parameter.

CHARSET UTF-8

```
DEFBSFILE ./dirdef/oracle.def PURGE CHARSET UTF-8  
USERID ggs_admin@S1E2, PASSWORD oracle  
TABLE osm$repapi.customers;  
TABLE osm$repapi.policies;
```

Columns mapping clauses

There are two techniques for handling columns mapping, using the parameters COLMAP or COLMATCH. Depending on how dissimilar the two tables one would be more preferred to use than the other.

The two methods are illustrated using the replicate process TS1E201.

COLMAP clause

Use COLMAP on cases where the columns have completely different names, or where one table has large number of column with only few columns differences. The benefit of COLMAP is using the USEDEFAULT option which causes implicit mapping of identical columns, followed by explicit column-to-column mapping process as illustrated next.

```
REPLICAT rs1e201  
SETENV (ORACLE_SID=T1C1)  
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)  
USERID ggs_admin@t1c1, PASSWORD oracle  
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE  
SOURCEDEFS /u01/app/oracle/ggs_2/dirdef/oracle.def  
MAP osm$repapi.customers, TARGET osm$repapi.customers, &  
COLMAP (USEDEFAULTS, cust_addr = cust_address);  
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

Notice, column mapping syntax allows the target column placed only on the left side of the assignment statement.

COLMATCH clause

Use COLMATCH on cases where tables have similar structure with different column names. COLMATCH clause uses either the NAMES or the SUFFIX/PREFIX options.

The NAMES option explicitly maps two columns names, while the SUFFIX/PREFIX relies on substring mapping to dynamically determine columns mapping. SUFFIX is used for ignoring columns suffix and PREFIX is used for ignoring column prefix.

The clause COLMATCH PREFIX PK_ maps any target column having a PK_ prefix to any source column of similar name. For example, PK_POL_NO with POL_NO.

COLMATCH using the NAMES option illustrated on the replicate parameter file.

```
REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@T1E2, PASSWORD oracle
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
COLMATCH NAMES CUST_ADDR = CUST_ADDRESS
MAP osm$repapi.customers, TARGET osm$repapi.customers;
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

Refer to chapter 10 to learn more about column mapping.

Built-in functions

Functions are mostly used for performing data transformation, required by the target database table structure and applications. Oracle GoldenGate supports categories of functions for characters, numbers, dates and general usages. The same principle of database and user-defined functions applies to Oracle GoldenGate functions in term of:

- Accepting of zero or more input parameters
- Allowing functions nesting
- Returning a value

Oracle GoldenGate functions invoked using the character '@' followed immediately by the function name. Functions are may be found in the parameter file of primary extract (capture), secondary extract (data pump) or replicate process.

Numbers and character functions

Table 1-4 lists number and character built-in functions.

Function Name	Description
@COMPUTE	Returns the result of arithmetic expressions into a target column. The return value is of type string
@NUMBIN	Converts a binary string into number
@NUMSTR	Converts a string to number
@STRCAT	Concatenates sequence of strings
@STRCMP	Compares two strings to determine whether they are equal, or the first string is less or greater than the second string.
@STREQ	Tests two strings to see if they equal or not. Return 1 for equal strings, returns 0 for not equal strings.
@STREXT	Returns substring from a string indicated by a begin and an end positions
@STRFIND	Returns the first occurrence of a substring within a string
@STRLEN	Returns the length of a string
@STRTRIM	Removes the leading spaces from a string
@STRNCAT	Concatenates one or more strings up to the specified number of characters per string
@STRNCMP	Compares two strings up to specific number of characters
@STRRTRIM	Trims the trailing spaces in a string
@STRSUB	Substitutes one string for another within a string
@STRTRIM	Trims both leading and trailing spaces in a string
@STRUP	Converts the string to uppercase

Table 1-4: Number and character built-in functions

Examples of using the character strings functions @STRUP, @STRFIND and @STRCAT follows.

- Converting the customer address to upper case for the target column.

```
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = @STRUP(cust_address));
```

- Validate the email address contains the '@' character:

```
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address,
        cust_email = @STRFIND (CUST_EMAIL, "@") > 0);
```

- Construct an upper case email by concatenating the name with specific domain. The upper case email achieved by nesting the two function @STRCAT and @STRUP.

```
MAP osm$repapi.customers, TARGET osm$repapi.customers,
COLMAP (USEDEFAULTS,
        cust_email=@STRUP(@STRCAT(cust_name,
                                   '@precisetrace.com')));
```

Date functions

Table 1-5 lists Oracle GoldenGate date related functions.

Function Name	Description
@DATE	Returns the date from specific input format, and output the date to specific output format
@DATEDIFF	Returns the difference between two dates
@DATENOW	Returns the current date and time value

Table 1-5: Oracle GoldenGate built-in date functions

An example of using the date function @DATENOW:

- Use the current date for the column CUST_NAME.

```
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS, &
        cust_addr = cust_address,
        cust_date = @DATENOW());
```

General functions

Table 1-6 lists Oracle GoldenGate general purpose functions.

Function Name	Description
@BINARY	Ensures source binary data remains binary in the target column when the source column is defined as a character column data type.
@BINTOHEX	Converts a binary string to hexadecimal string value
@GETENV	Returns Oracle GoldenGate environment variable value.
@GETVAL	Extracts parameter value from a stored procedure when using

	SQLEXEC clause, requires the name procedure and parameter name.
@HEXTOBIN	Converts from hexadecimal to binary string value
@RANGE	Used to provide parallel processing for enabling multiple secondary extract or replicate processes read the same trail.
@TOKEN	Used to retrieves environment variable value for column mapping.

Table 1-6: Oracle GoldenGate general purpose built-in functions

Example of using general function @TOKEN:

- Retrieve Oracle GoldenGate environment variable for hostname, operating system user and database name.

```
MAP osm$repapi.customers, TARGET osm$repapi.customers,
COLMAP (USEDEFAULTS, &
        cust_addr = cust_address,
        cust_date = @DATENOW(),
        cust_hostname = @TOKEN ('TK_HOST'),
        cust_osname   = @TOKEN ('TK_OSUSER'),
        cust_dbname   = @TOKEN ('TK_DBNAME'));
```

Data selection and filtering

Data selection and filtering are fundamentals to data integration, Oracle GoldenGate provides the capabilities to perform data selection and filtering to meet applications requirements. There are several advantages for performing data selection and filtering from within GoldenGate, the most appealing is data security.

Data security

Implemented as a form of physical private database (PPD), where data routing to the target databases may be designed based on application context. A sophisticated use cases is implementing physical private database based on partitioned source table. Figure 1-27 illustrates limiting data access by implementing physical private database based on the table partitioning design.

The CUSTOMERS table is partitioned by range. The partitioning column is CUST_NO created as shown below.

```
SQL> CREATE TABLE customers (
2  cust_no      NUMBER NOT NULL PRIMARY KEY,
3  cust_name    VARCHAR2(128) NOT NULL,
```

```

4  cust_email    VARCHAR2(128) NOT NULL,
5  cust_mobile   NUMBER,
6  cust_address  VARCHAR2(128))
7  PARTITION BY RANGE (cust_no)
8    (PARTITION p100000 VALUES LESS THAN (100001),
9     PARTITION p200000 VALUES LESS THAN (200001),
10    PARTITION p300000 VALUES LESS THAN (300001),
11    PARTITION p999999 VALUES LESS THAN (MAXVALUE)) ;

```

Table created.

SQL>

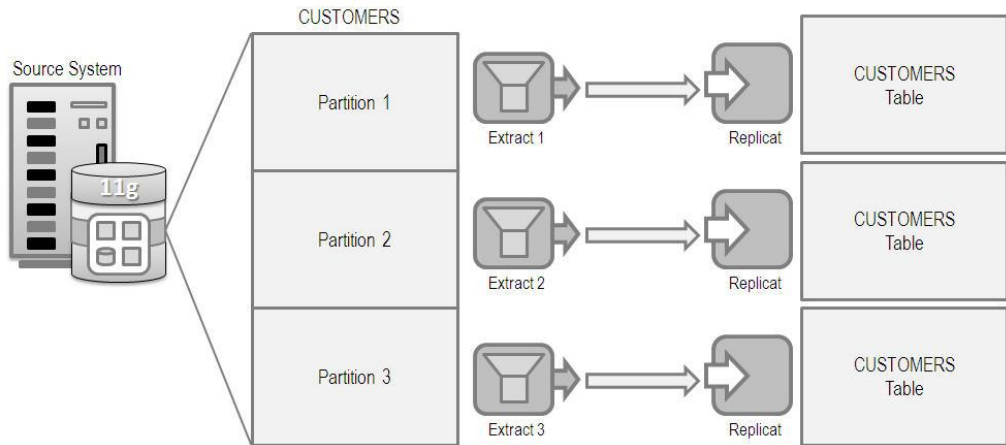


Figure 1-27: Physical private database (PPD)

For each partition, a dedicated primary extract performs change data capture using selection clause, creating its own local trail. Next, a data pump extract routes the data to a data server collector which creates a remote trail. The replicate process applies the transactions to the target database CUSTOMERS table.

The above example is more realistic when implemented using interval or list partitioning methods. The combination of Oracle GoldenGate selection, filtering, tokens and application context contribute for building secure data model.

Data selection

Oracle GoldenGate selection is conceptually similar to the WHERE clause found on SQL. The purpose is to limit data replication to specific criteria. The selection can be developed as part of the primary extract, data pump extract or replicate process. The

following illustrates data selection on the CUSTOMERS and POLICIES tables for customers with policy no greater than 100000 through the primary extract ES1E201.

```
EXTRACT es1e201
SETENV (ORACLE_SID=S1E2)
SETENV (ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD oracle
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers, WHERE (cust_no > 100000);
TABLE osm$repapi.policies, WHERE (cust_no > 100000);
```

The selection clause should be defined for all related tables to avoid referential integrity violation when applying the transactions to the target database. Hence, it's necessary to construct the where clause to the CUSTOMERS table and POLICIES table as well.

Data filtering

Data filtering is suitable for complex data selection by utilizing Oracle GoldenGate built-in functions. For the replicate process, the FILTER parameter is developed as part of the MAP statement.

The example below illustrates the use of the FILTER parameter with the function @STRFIND to filter out email addresses that does not contains the character '@'.

```
REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@T1C1, PASSWORD oracle
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address),
FILTER (@STRFIND (cust_email, "@") > 0);
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

Data transformation

The need to transform tables columns values is very essentials for implementing an online operational data warehouse repository for business intelligence applications.

Data transformation predominately applies to:

- Heterogeneous environment requirements
- The target database application uses rigid data types of different format
- Data warehouse repository summaries

For more complex operational data store (ODS), one form of data warehouse repository, data transformation is often essential. The transformation may be configured as part of the primary extract, secondary extract, the delivery process or a combination of any of them. Transformations achieved using Oracle GoldenGate offload the database and enable transparent application support . A simple yet common transformation example is the conversion of number data type to formatted character string data type as shown below:

Source table column data type: NUMBER
Source input value: 5085797993

Target table column data type: VARCHAR2(16)
Target output value: (508) 579-7993

Nesting Oracle GoldenGate built-in functions @STRCAT, @STREXT and @STRNUM convert the number to a formatted string. Starting with @STRNUM to convert the number to string, @STREXT to extract substrings of input string and then use @STRCAT to concatenate the substrings.

This data transformation is developed as part of replicate process configuration. Because of mismatched column names for the customer address, column mapping is also included.

```

REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@T1C1, PASSWORD oracle
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address,
        cust_mobile = @STRCAT('(',
        @STREXT(@STRNUM(cust_mobile), 1, 3), ')',
        ' ',
        @STREXT(@STRNUM(cust_mobile), 4, 6),
        '-',
        @STREXT(@STRNUM(cust_mobile), 7, 10)));
MAP osm$repapi.policies, TARGET osm$repapi.policies;

```

Tokens

Oracle GoldenGate tokens are attributes to stores environment variable values, which are available on the header of trail record header. Using Oracle GoldenGate built-in function @TOKEN within the parameter files retrieves the token value.

Table 1-6 lists pre-defined Oracle GoldenGate tokens.

Token Name	Description
TK_HOST	Stores the host name
TK_GROUP	Stores the operating system group name
TK_OSUSER	Stores the operating system user name
TK_DOMAIN	Stores the hostname domain
TK_COMMIT_TS	Stores the current commit transaction sequence number
TK_POS	Stores the current read position
TK_RBA	Stores the relative byte address (RBA)
TK_TABLE	Stores the table name
TK_OPTYPE	Stores the operation types (INSERT, UPDATE, DELETE or DDL)

Table 1-6: Oracle GoldenGate pre-defined tokens

Another type of tokens are user-token. These are defined on the source system and moved to target system for event handling, providing extendibility for attributes that are not pre-defined by Oracle GoldenGate.

The replicate RS1E201 parameter file illustrates a basic implementation of token by associating each row with Oracle GoldenGate environment pre-defined tokens values that can be used by other applications, such as auditing.

```
T1C1>DESC customers
Name                                     Null?    Type
-----
CUST_NO                                NOT NULL NUMBER
CUST_NAME                              NOT NULL VARCHAR2 (128)
CUST_EMAIL                             NOT NULL VARCHAR2 (400)
CUST_MOBILE                             NOT NULL VARCHAR2 (16)
CUST_ADDR                               VARCHAR2 (128)
HOST                                   VARCHAR2 (128)
GG_GROUP                               VARCHAR2 (128)
OSUSER                                 VARCHAR2 (128)
DOMAIN                                 VARCHAR2 (128)
BA_IND                                 VARCHAR2 (128)
```

COMMIT_TS	VARCHAR2 (128)
POS	VARCHAR2 (128)
RBA	VARCHAR2 (128)
TABLERNAME	VARCHAR2 (128)
OPTYPE	VARCHAR2 (128)

```

REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@T1C1, PASSWORD oracle
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address,
        host      = @TOKEN ('TK_HOST'),
        gg_group  = @TOKEN ('TK_GROUP'),
        osuser    = @TOKEN ('TK_OSUSER'),
        domain    = @TOKEN ('TK_DOMAIN'),
        ba_ind    = @TOKEN ('TK_BA_IND'),
        commit_ts = @TOKEN ('TK_COMMIT_TS'),
        pos       = @TOKEN ('TK_POS'),
        rba       = @TOKEN ('TK_RBA'),
        tablename = @TOKEN ('TK_TABLE'),
        optype    = @TOKEN ('TK_OPTYPE'));
MAP osm$repapi.policies, TARGET osm$repapi.policies;

```

Refer to chapter 10 to learn about advanced implementation of how to store environment variable into Oracle database instance memory, SGA.

Programs call-out

Oracle GoldenGate extends its functionalities through call-out to database stored procedures and C programs. For database stored procedures, the call-out enabled through SQLEXEC parameter, call-out to C programs developed using user-exist.

To invoke a database stored procedures, extends the TARGET clause to include the SQLEXEC option. The parameter SPNAME specifies the <owner>.<procedure name> and PARAMS specifies the list of formal parameters.

```

MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address),
SQLEXEC (SPNAME osm$repapi.load_cust_pic,

```

```
PARAMS (pcust_no = cust_no));
```

Refer to chapter 10 for an end-to-end demonstration on how to invoke database stored procedure from within Oracle GoldenGate. The demonstration uses of DBMS_FILE_TRANSFER Oracle database built-in package to enable moving an image file of type BFILE from source system to target system. The purpose of the demonstration to show the technique for handling unsupported data types such as BFILE.

Configuration options

Post-deployment of Oracle GoldenGate instance often requires environment enhancements. This section is limited to the basic enhancements only.

We will discuss enabling the following:

- Secure parameter files
- Data compression
- Scheduler
- Batch SQL

Secure parameter files

Extract and replicate processes parameter files often requires login to the database. Best practices should avoid using clear-text password, instead use encrypted password. Two methods are available: default method and named method.

We will illustrate using the default method to encrypt and use the encrypted password.

- From the GGSCI command-line, run the ENCRYPT command using the default algorithm, BLOWFISH.

```
GGSCI (ggs-source) 34> ENCRYPT PASSWORD oracle, ENCRYPTKEY default
```

```
Using Blowfish encryption with DEFAULT key.
```

```
Encrypted password: AACAAAAAAAAAAAAAGAIFAAUDVHCFUGFIYF
```

```
Algorithm used: BLOWFISH
```

```
GGSCI (ggs-source) 35>
```

- Test the encrypted password

```
GGSCI (ggs-source) 39> DBLOGIN USERID ggs_admin@S1E2,
PASSWORD AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF, ENCRYPTKEY
DEFAULT
```

Successfully logged into database.

```
GGSCI (ggs-source) 36>
```

- Copy the encrypted password into the extract and replicate parameter files.

```
EXTRACT es1e201
SETENV (ORACLE_SID=S1E2)
SETENV (ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD
AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF, ENCRYPTKEY DEFAULT
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers, WHERE (cust_no > 100000);
TABLE osm$repapi.policies, WHERE (cust_no > 100000);
```

Data compression

Moving data between source and target systems highly benefits from trail files compression, the default is non-compressed. Compression is transparent and enabled by adding the compression parameter option to the data pump process.

The following illustrates using Oracle GoldenGate compression option of type zlib. Automatic decompression is performed by the data collector server.

```
EXTRACT epmp01
USERID ggs_admin@S1E2, PASSWORD oracle
PASSTHRU
RMTHOST localhost, MGRPORT 7810, COMPRESS, &
COMPRESSTHRESHOLD 512
RMTRAIL /u01/app/oracle/ggs_2/dirdat/ta
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

Job scheduler

Oracle GoldenGate has its own job scheduler sub-system. It enables executing SQL statements and PL/SQL stored sub-programs. However, for complex event scheduling, consider using the database scheduler DBMS_SCHEDULER API. Oracle GoldenGate scheduler uses simplified expression to trigger the jobs.

An application area for using Oracle GoldenGate scheduler is to implement self-monitoring tool by scheduling the replicate processes to send an email every 10 minutes to inform the Oracle GoldenGate administrator it's on RUNNING status.

```
SQL> CREATE PROCEDURE email_rep_status(p1 VARCHAR2) IS
  2 BEGIN
  3     smtp_send(p1||' IS RUNNING...');
  4 END;
  5 /
```

Procedure created.

SQL>

Now, add the parameter SQLEXEC to each replicate process parameter file. Stop and restart replicate processes.

```
REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1
)
SETENV (NLS_LANG=AMERICAN AMERICA.AL32UTF8)
USERID ggs_admin@T1C1, PASSWORD oracle
SQLEXEC "CALL email_rep_status('RS1E201')" EVERY 10 MINUTES
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address),
SQLEXEC (SPNAME osm$repapi.load_cust_pic,
        PARAMS (pcust_no = cust_no));
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

An informational message is sent to the process report file whenever every time the statement executed. Nice and simple tool!

```
2013-07-14 22:23:25 INFO      OGG-00888  SQL statement
executed successfully on interval.
```

Batch SQL

The replicate process parameter BATCHSQL increases the apply throughput by classifying and building in-memory arrays for similar SQL statements to be applied as one database operation.

A typical environment is an OLTP system that applies transactions differing on bind variable values only. Oracle GoldenGate supercharges the replicate processes using batch SQL technique. By default, replicate processes apply transactions one at a time, with BATCHSQL the replicate process applies transactions on batch mode.

The replicate process parameter BATCHSQL follows.

```
REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@T1C1, PASSWORD oracle
BATCHSQL
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
SOURCEDEFS ./dirdef/oracle.def
MAP osm$repapi.customers, TARGET osm$repapi.customers, &
COLMAP (USEDEFAULTS,
        cust_addr = cust_address);
MAP osm$repapi.policies, TARGET osm$repapi.policies;
```

Refer to chapter 13 to learn more about BATCHSQL settings.

Automatic storage management (ASM) support

Oracle Database 10g release 2 introduced Oracle automatic storage management (ASM), since then Oracle corporation has been enhancing ASM to provides high-performance file system management supporting all Oracle database file types. Figure 1-28 depicts Oracle ASM instance showing the data flow from Oracle GoldenGate extract on an ASM environment.

Oracle GoldenGate change data capture requires ASM authentication for accessing the transaction logs. The extract parameter TRANLOGOPTIONS handles authentication to the ASM instance.

```
TRANLOGOPTIONS ASMUSER SYS@<ASM_instance>, ASMPASSWORD
<password> [<algorithm> ENCRYPTKEY {<keyname> | DEFAULT}]
```

The parameter TRANLOGOPTIONS is database type dependent, and may appear several times within the same parameter file.

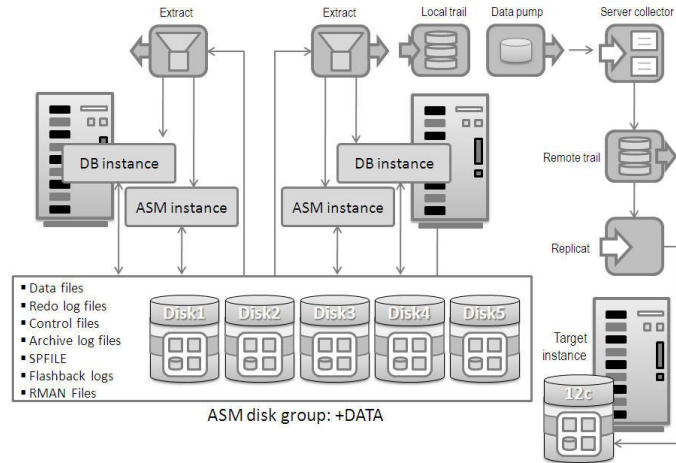


Figure 1-28: Oracle ASM for GoldenGate

To enable Oracle GoldenGate capture from an ASM instance transaction logs, it requires configuring the network for the ASM instance. The steps to prepare Oracle GoldenGate for an ASM instance follows.

Step 1: Ensure the database listener is configured to handle connection, and the TNS alias is configured. The network configuration is for the grid infrastructure network.

```
SID_LIST_LISTENER =
(
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = S2E2.precisetrace.com)
      (ORACLE_HOME = /u01/app/oracle/product/11.2.0/dbhome_1)
      (SID_NAME = S2E2)
    )
    (SID_DESC =
      (ORACLE_HOME = /u01/app/oracle/product/11.2.0/grid)
      (SID = +ASM)
    )
  )
)

LISTENER =
(DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP) (HOST = ggs-source) (PORT = 1521))
)

ADR_BASE_LISTENER = /u01/app/oracle

ASM =
(DESCRIPTION =
  (ADDRESS_LIST =
```

```

        (ADDRESS = (PROTOCOL = TCP) (HOST = ggs-source) (PORT = 1521))
    )
    (CONNECT_DATA =
        (SID = +ASM)
    )
)

```

Step 2: Test connectivity using SQL*Plus as SYSASM users. The test must be successful before proceeding to step 3.

```

$ . oraenv
ORACLE_SID = [S2E2] ? +ASM
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus /nolog

SQL*Plus: Release 11.2.0.3.0 Production on Fri May 31 11:08:35 2013

Copyright (c) 1982, 2011, Oracle. All rights reserved.

SQL> conn / as SYSASM
Connected.
SQL>

```

Step 3: Modify the primary extract parameter file to include the TRANSLOGOPTIONS parameter.

```

EXTRACT es1e201
SETENV (ORACLE_SID=S1E2)
SETENV (ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF,
ENCRYPTKEY DEFAULT
TRANLOGOPTIONS ASMUSER sys@ASM, ASMPASSWORD oracle_4U
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers, WHERE (cust_no > 100000);
TABLE osm$repapi.policies, WHERE (cust_no > 100000);

```

When using Oracle GoldenGate on an ASM instance, ensure the maximum number of concurrently processes parameter is sufficient to handle the addition of GoldenGate extracts, default value is 50. When using Oracle GoldenGate parallel processing, set the value to support the total number of processes defined by the @RANGE function. A typical parameter value setting is 300.

Parallel processing

Oracle GoldenGate supports high-performance parallel processing. The degree of parallelism controlled using the @RANGE function. Figure 1-29 delivers 1-3-6 degree of parallelism. The extract ES1E201 perform change data capture from the list

of tables. The data pump extract groups EPMP01, EPMP02, and EPMP03 parallelize extracting from the local trail `./dirdat/sa` into degree of 3, each data pump extract has its associated data server collector, which in turn re-assembles the data and writes to its own remote trail. Ultimately, using parallel processing maintains stable sub-second latency and lagging.

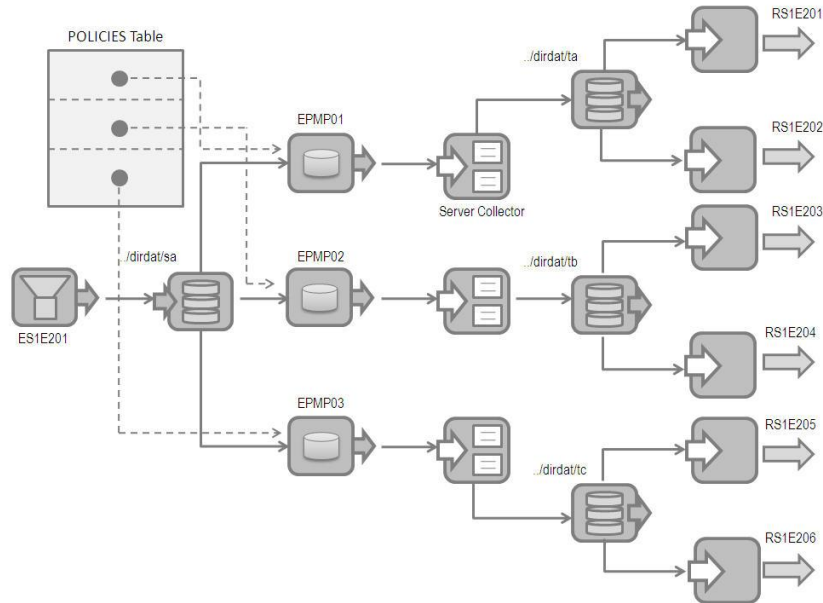


Figure 1-29: Oracle GoldenGate Parallel processing

The replicate groups RS1E201, RS1E202, RS1E203, RS1E204, RS1E205 and RS1E206 parallelize its associated remote trail extract into degree of 2, hence a total of 6 replicates processes are used to apply the transaction, captured by the extract ES1E201, to the target database.

Implement parallelism

Parallel processing is a key for maximizing system scalability and throughput. Oracle GoldenGate built-in function `@RANGE` implemented to deploy parallel processing for increasing throughput on large volume transaction tables for online synchronization, keeping the lagging minimum.

Specify each range in a **FILTER** clause in a **TABLE** or **MAP** statement. `@RANGE` function is very similar to hash database partitioning, it's safe and scalable preserving data integrity by guaranteeing that the row is processed by related process groups.

@RANGE computes a hash value of the columns specified in the input. If no columns are specified, the KEYCOLS clause of the TABLE or MAP statement is used to determine the columns to hash, if a KEYCOLS clause exists. Otherwise, the primary key columns are used, which the default.

The function takes three arguments, the current process number, the degree of parallelism and partitioning column(s), which optional. The general syntax:

```
@RANGE (<range>, <total ranges> [, <column>] [, <column>] [, ...])
```

Memory resource requirement

Prior to deploying high-degree parallel processing, calculate and provision the memory requirements for all processes. The required memory size is dynamic and depends on the transaction volume processed by each process. Table 1-x shows the calculation for the above parallel processing design using the operating system Linux process status (ps) and process mapping (pmap) commands. Table 1-x illustrates the memory consumption for the above design while the source and target system engaged on real-time data replication for a low volume transaction.

Start by determining the Linux process ID, then use the process for on pmap (process map) operating system command.

```
GGSCI (ggs-target) 28> SHELL ps -edf | grep RS1E201

oracle      6427   6584    0 05:25 pts/3      00:00:00 sh -c ps -edf |
grep RS1E201
oracle      6429   6427    0 05:25 pts/3      00:00:00 grep RS1E201
oracle      8943   4964    0 04:22 ?          00:00:03
/u01/app/oracle/ggs/replicat PARAMFILE
/u01/app/oracle/ggs/dirprm/rs1e201.prm REPORTFILE
/u01/app/oracle/ggs/dirrpt/RS1E201.rpt PROCESSID RS1E201 USESUBDIRS

GGSCI (ggs-target) 29> shell pmap -x 6427 | grep total

total kB    204252      -      -      -

GGSCI (ggs-target) 30>
```

Follow the above syntax to calculate the memory for all processes as shown in table 1-7.

GoldenGate process Name	GoldenGate process Type	Memory size (KB)
----------------------------	----------------------------	---------------------

ES1E201	Primary extract	204252
EPMP01	Second extract, data pump	112396
EPMP02	Second extract, data pump	111808
EPMP03	Second extract, data pump	111812
RS1E201	Replicate process	131476
RS1E202	Replicate process	131476
RS1E203	Replicate process	131128
RS1E204	Replicate process	131124
RS1E205	Replicate process	131128
RS1E206	Replicate process	131476
Total (GB)		1.27

Table 1-7: Oracle GoldenGate process memory utilization

Parallel processing implementation

The following shows the parameter files and the GGSCI ADD EXTRACT, ADD EXTTRAIL and ADD RMTTRAIL commands executed to deploy the above parallel processing design shown in figure 1-30.

The source system setup creates the primary extract group ES1E201 and the secondary extract groups EPMP01, EPMP02 and EPMP03. The parameter files are listed below. Because of the FILTER clause, the parameter NOPASSTHRU is required.

-- es1e201.prm

```
EXTRACT es1e201
SETENV (ORACLE_SID=S1E2)
SETENV (ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF,
ENCRYPTKEY DEFAULT
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

-- epmp01.prm

```
EXTRACT epmp01
USERID ggs_admin@S1E2, PASSWORD AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF,
ENCRYPTKEY DEFAULT
NOPASSTHRU
RMTHOST ggs-target, MGRPORT 7812
RMTTRAIL ./dirdat/ta
TABLE osm$repapi.customers, FILTER (@RANGE(1, 3));
TABLE osm$repapi.policies, FILTER (@RANGE(1, 3));
```

```
-- epmp02.prm
```

```
EXTRACT epmp02
USERID ggs_admin@S1E2, PASSWORD AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF,
ENCRYPTKEY DEFAULT
NOPASSTHRU
RMTHOST ggs-target, MGRPORT 7812
RMTTRAIL ./dirdat/tb
TABLE osm$repapi.customers, FILTER (@RANGE(2, 3));
TABLE osm$repapi.policies, FILTER (@RANGE(2, 3));
```

```
-- epmp03.prm
```

```
EXTRACT epmp03
USERID ggs_admin@S1E2, PASSWORD AACAAAAAAAAAAGAIFAAUDVHCFUGFIYF,
ENCRYPTKEY DEFAULT
NOPASSTHRU
RMTHOST ggs-target, MGRPORT 7812
RMTTRAIL ./dirdat/tc
TABLE osm$repapi.customers, FILTER (@RANGE(3, 3));
TABLE osm$repapi.policies, FILTER (@RANGE(3, 3));
```

Now, run the ADD EXTRACT, ADD EXT*TRAIL and ADD RMT*TRAIL commands.

```
GGSCI (ggs-source) 5> ADD EXTRACT es1e201, TRANLOG, BEGIN
now, THREADS 1
EXTRACT added.
```

```
GGSCI (ggs-source) 6> ADD EXTTRAIL ./dirdat/sa, EXTRACT
es1e201, MEGABYTES 32
EXTTRAIL added.
```

```
GGSCI (ggs-source) 7> ADD EXTRACT epmp01, EXTTRAILSOURCE
./dirdat/sa
EXTRACT added.
```

```
GGSCI (ggs-source) 8> ADD RMTTRAIL ./dirdat/ta, EXTRACT
epmp01, MEGABYTES 32
RMTTRAIL added.
```

```
GGSCI (ggs-source) 9> ADD EXTRACT epmp02, EXTTRAILSOURCE
./dirdat/sa
EXTRACT added.
```

```
GGSCI (ggs-source) 10> ADD RMTTRAIL ./dirdat/tb, EXTRACT
epmp02, MEGABYTES 32
RMTTRAIL added.
```

```
GGSCI (ggs-source) 11> ADD EXTRACT epmp03, EXTTRAILSOURCE
./dirdat/sa
EXTRACT added.

GGSCI (ggs-source) 12> ADD RMTTRAIL ./dirdat/tc, EXTRACT
epmp03, MEGABYTES 32
RMTTRAIL added.
```

The target system setup creates the replicate process RS1E201, RS1E201, RS1E201, RS1E201, RS1E201 and RS1E201. The parameter files prepared to successfully start the replicate processes.

-- rs1e201.prm

```
REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@tlc1, PASSWORD oracle
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers,
FILTER (@RANGE(1, 2));
MAP osm$repapi.policies, TARGET osm$repapi.policies,
FILTER (@RANGE(1, 2));
```

-- rs1e202.prm

```
REPLICAT rs1e202
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@tlc1, PASSWORD oracle
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e202.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers,
FILTER (@RANGE(2, 2));
MAP osm$repapi.policies, TARGET osm$repapi.policies,
FILTER (@RANGE(2, 2));
```

-- rs1e203.prm

```
REPLICAT rs1e203
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@tlc1, PASSWORD oracle
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e203.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers,
FILTER (@RANGE(1, 2));
MAP osm$repapi.policies, TARGET osm$repapi.policies,
FILTER (@RANGE(1, 2));
```

```
-- rs1e204.prm
```

```
REPLICAT rs1e204
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@t1c1, PASSWORD oracle
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e204.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers,
FILTER (@RANGE(2, 2));
MAP osm$repapi.policies, TARGET osm$repapi.policies,
FILTER (@RANGE(2, 2));
```

```
-- rs1e205.prm
```

```
REPLICAT rs1e205
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@t1c1, PASSWORD oracle
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e205.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers,
FILTER (@RANGE(1, 2));
MAP osm$repapi.policies, TARGET osm$repapi.policies,
FILTER (@RANGE(1, 2));
```

```
-- rs1e206.prm
```

```
REPLICAT rs1e206
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@t1c1, PASSWORD oracle
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e206.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers,
FILTER (@RANGE(2, 2));
MAP osm$repapi.policies, TARGET osm$repapi.policies,
FILTER (@RANGE(2, 2));
```

Let's start by creating the checkpoint table. This is the default checkpoint location for replicate processes. The ADD CHECKPOINTTABLE command requires successful login to the database.

```
GGSCI (ggs-target) 2> DBLOGIN USERID ggs_admin@t1c1, PASSWORD
oracle
Successfully logged into database.

GGSCI (ggs-target) 3> ADD CHECKPOINTTABLE
```

```
No checkpoint table specified. Using GLOBALS specification  
(ggs_admin.chkpttab)...
```

```
Successfully created checkpoint table ggs_admin.chkpttab.
```

```
GGSCI (ggs-target) 4>
```

Now, creating replicate processes RS1E201, RS1E202, RS1E203, RS1E204, RS1E205 and RS1E206.

```
GGSCI (ggs-target) 6> ADD REPLICAT rs1e201, EXTTRAIL  
./dirdat/ta  
REPLICAT added.
```

```
GGSCI (ggs-target) 7> ADD REPLICAT rs1e202, EXTTRAIL  
./dirdat/ta  
REPLICAT added.
```

```
GGSCI (ggs-target) 8> ADD REPLICAT rs1e203, EXTTRAIL  
./dirdat/tb  
REPLICAT added.
```

```
GGSCI (ggs-target) 9> ADD REPLICAT rs1e204, EXTTRAIL  
./dirdat/tb  
REPLICAT added.
```

```
GGSCI (ggs-target) 10> ADD REPLICAT rs1e205, EXTTRAIL  
./dirdat/tc  
REPLICAT added.
```

```
GGSCI (ggs-target) 11> ADD REPLICAT rs1e206, EXTTRAIL  
./dirdat/tc  
REPLICAT added.
```

Start and verify the source system processes status

```
GGSCI (ggs-source) 40> START er *
```

```
Sending START request to MANAGER ...  
EXTRACT EPMP01 starting
```

```
Sending START request to MANAGER ...  
EXTRACT EPMP02 starting
```

```
Sending START request to MANAGER ...  
EXTRACT EPMP03 starting
```

```
Sending START request to MANAGER ...
EXTRACT ES1E201 starting
```

```
GGSCI (ggs-source) 41> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	RUNNING	EPMP01	00:00:00	00:00:12
EXTRACT	RUNNING	EPMP02	00:00:00	00:00:11
EXTRACT	RUNNING	EPMP03	00:00:00	00:00:10
EXTRACT	RUNNING	ES1E201	00:00:00	00:00:09

```
GGSCI (ggs-source) 43>
```

Start and verify the target system processes status.

```
GGSCI (ggs-target) 11> START ER *
```

```
Sending START request to MANAGER ...
REPLICAT RS1E201 starting
```

```
Sending START request to MANAGER ...
REPLICAT RS1E202 starting
```

```
Sending START request to MANAGER ...
REPLICAT RS1E203 starting
```

```
Sending START request to MANAGER ...
REPLICAT RS1E204 starting
```

```
Sending START request to MANAGER ...
REPLICAT RS1E205 starting
```

```
Sending START request to MANAGER ...
REPLICAT RS1E206 starting
```

```
GGSCI (ggs-target) 12> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RS1E201	00:00:00	00:00:09
REPLICAT	RUNNING	RS1E202	00:00:00	00:00:09
REPLICAT	RUNNING	RS1E203	00:00:00	00:00:08
REPLICAT	RUNNING	RS1E204	00:00:00	00:00:08
REPLICAT	RUNNING	RS1E205	00:00:00	00:00:08
REPLICAT	RUNNING	RS1E206	00:00:00	00:00:08

Generating Reports

Oracle GoldenGate generates various types of reports for troubleshooting and monitoring purposes. These report are available from GGSCI command-line.

Use the view GGSCI command to view process report.

```
GGSCI (ggs-target) 13> VIEW REPORT rs1e201
```

GGSCI STATS command reports process statistics. For example, display the @RANGE statistics using wild-card character. The STATS indicates parallel processing has been distributed almost evenly across the replicate processes.

```
GGSCI (ggs-target) 15> STATS REPLICAT rs1e2*
```

Sending STATS request to REPLICAT RS1E201 ...

Start of Statistics at 2013-11-05 08:52:00.

Replicating from OSM\$REPAPI.CUSTOMERS to OSM\$REPAPI.CUSTOMERS:

```
*** Total statistics since 2013-11-05 08:23:19 ***
Total inserts                19494.00
Total updates                 0.00
Total deletes                 0.00
Total discards                0.00
Total operations              19494.00

*** Daily statistics since 2013-11-05 08:23:19 ***
Total inserts                19494.00
Total updates                 0.00
Total deletes                 0.00
Total discards                0.00
Total operations              19494.00

*** Hourly statistics since 2013-11-05 08:23:19 ***
Total inserts                19494.00
Total updates                 0.00
Total deletes                 0.00
Total discards                0.00
Total operations              19494.00

*** Latest statistics since 2013-11-05 08:23:19 ***
Total inserts                19494.00
Total updates                 0.00
Total deletes                 0.00
```

Total discards	0.00
Total operations	19494.00

Monitoring processes lagging

The difference between the commit and the apply determines the amount of lagging. The following show the lag for replicate process RS1E201.

```
GGSCI (ggs-target) 16> LAG REPLICAT rs1e201
```

```
Sending GETLAG request to REPLICAT RS1E201 ...
```

```
Last record lag: 12 seconds.
```

```
At EOF, no more records to process.
```

The column Lag at Chkpt column of the INFO ALL command display the lagging of the replicate process.

```
GGSCI (ggs-target) 17> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RS1E201	00:00:00	00:00:07
REPLICAT	RUNNING	RS1E202	00:00:00	00:00:07
REPLICAT	RUNNING	RS1E203	00:00:00	00:00:06
REPLICAT	RUNNING	RS1E204	00:00:00	00:00:06
REPLICAT	RUNNING	RS1E205	00:00:00	00:00:06
REPLICAT	RUNNING	RS1E206	00:00:00	00:00:06

```
GGSCI (ggs-target) 18>
```

Events logging system

Oracle GoldenGate is integrated with Linux/UNIX syslog messaging system and Windows Event Viewer for events logging. There are three files for managing runtime errors, namely: ggserr.log, tcperr and ggMessage.dat all located on Oracle GoldenGate home directory. The file ggserr.log is for logging currently running instance activities, tcperr defines rules for handling TCP/IP errors and actions and ggMessage.dat is an obfuscated data file for storing Oracle GoldenGate error messages specific to the instance version.

By default Oracle GoldenGate writes all levels of messages to ggserr.log file, which is crucial for newly deployed instance for monitoring. After the instance runs operational and highly stable, logging level can be adjusted to filter out less important messages send to the log file. The logging level is determined by Oracle GoldenGate SYSLOG parameter. Setting SYSLOG at GLOBALS parameter file allows Oracle

GoldenGate to write and maybe filter out messages for all instance processes, setting SYSLOG at mgr.prm parameter files allows Oracle GoldenGate to write and maybe filter out the manager process related messages only. Setting SYSLOG on both locations causes the manager to override the GLOBALS setting for the manager process, this allows the use of separate settings between the manager and other processes. SYSLOG parameter takes the following arguments for level settings.

ALL

Writes all Oracle GoldenGate messages. This setting is the same as combining INFO, WARN, ERROR arguments. This is the default level of logging messages. This argument cannot be combined with other arguments.

NONE

Suppresses all Oracle GoldenGate messages. This is not a recommended setting and it should be used for non-production environment only. This argument cannot be combined with other arguments.

INFO

Writes messages of type INFO (information) only. This argument can be combined with WARN (warning) and ERROR arguments. Information messages include activities such as stopping and starting extracts and replicates processes.

WARN

Write messages of type WARN (warning). This argument can be combined with INFO (information) and ERROR arguments.

ERROR

Write messages of type ERROR. This argument can be combined with INFO (information) and WARN (warning) arguments.

Because the log file ggserr.log is not cyclic, periodically archiving the log file is recommend to avoid the file growing linearly. When ggserr.log is deleted, Oracle GoldenGate re-creates the logging file ggserr.log once it's trying to write the very next message. The following Linux/UNIX shell program archives ggserr.log file to /tmp directory when ggserr.log exceeds 10MB (10485760 bytes).

```
#-----  
#--      DISCLAIMER:
```

```

#--
#--      This script is provided for educational purposes only.
#--      The script has been tested on an Oracle GoldenGate instance
#--      and run as intended. However; prior to using the script on
#--      production systems, a dynamic test should be conducted prior
#--      to deployment.
#--
#--      Copyright (c) Rampant TechPress. 2014. All Rights Reserved.
#-----
#
#!/bin/bash
GGS_HOME="/u01/app/ogg/11.2.0"
if [ -f $GGS_HOME/ggserr.log ]; then
    FS=$( stat -c %s $GGS_HOME/ggserr.log)
    if [ $FS -gt 10485760 ]; then
        mv $GGS_HOME/ggserr.log /tmp
        echo "$GGS_HOME/ggserr.log was last archived on:" `date` >>
/tmp/ggserr.txt
    fi
else
    echo "$GGS_HOME/ggserr.log does not exist:" `date` >>
/tmp/ggserr.txt
fi

```

For scheduled archiving on Linux/UNIX platforms, save the above shell program to a file, for example ggserr.sh, set the execute permission, add the file ggserr.sh to /etc/crontab using the command crontab. However; Adapting Oracle GoldenGate monitor notification system enables immediate response to all types events.

The Syntax for SYSLOG parameter is shown next. The default setting is SYSLOG ALL, which writes INFO (information), WARN (warning) and ERROR messages.

```
SYSLOG {[ALL | NONE] | [, INFO] [, WARN] [, ERROR]}
```

Windows Event Viewer

Oracle GoldenGate on MS Windows for SQL Server writes GoldenGate messages to ggserr.log file and to Windows Event Viewer, if enabled. The event levels is controlled by setting the parameter SYSLOG as explained above. To sort Oracle GoldenGate related messages, from Administrative Events click on the source column and scroll down to GGS ER.

Oracle GoldenGate is enabled to utilize Windows Event Viewer by running Oracle GoldenGate install program with the addevents option, this is normally performed after the installation. Then from Oracle GoldenGate home directory, copy the files category.dll and ggsmmsg.dll libraries to Windows SYSTEM32 directory for more

detailed Windows specific messages. Figure 1-30 shows an ERROR level message from Windows Event Viewer, the same message is written to ggserr.log file.

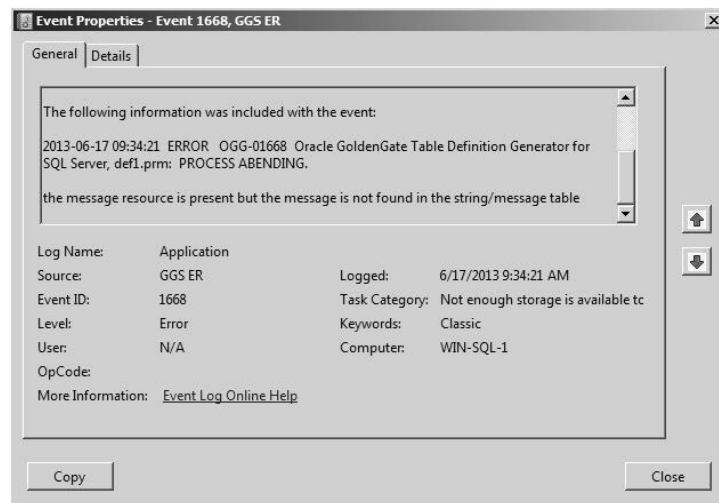


Figure 1-30: MS Windows event viewer for Oracle GoldenGate

Windows events handling

MS Windows allows you to attach a task to an event level, providing an automated event handling for Oracle GoldenGate log messages. For example, events of level Information and Warning can be regularly cleared out to avoid the Event Viewer from growing quickly, events of level Error are associated with an action for immediate notification and action, as explained next.

Right-clicking on the event to start the Create Basic Task Wizard. Figure 1-31 associate an event of level Error with the action Send an email. Send an E-mail task uses the inputs: From, To, Subject, Text, Attachment, and SMTP server. An alternative to using the Create Basic Task Wizard is to create a customized archiving batch program to run using the Windows scheduler created by the AT command. AT command schedules operating system commands and batch programs (utilities) to run at a specific time and date, the computename argument refer to the local host (default) or remote host. Before using the AT command, Windows schedule service must be running.

Failure to manage ggserr.log file and Windows Event Viewer may result on:

- Performance related issues for large file size of ggserr.log
- Disk storage related issues for excessive events messaging

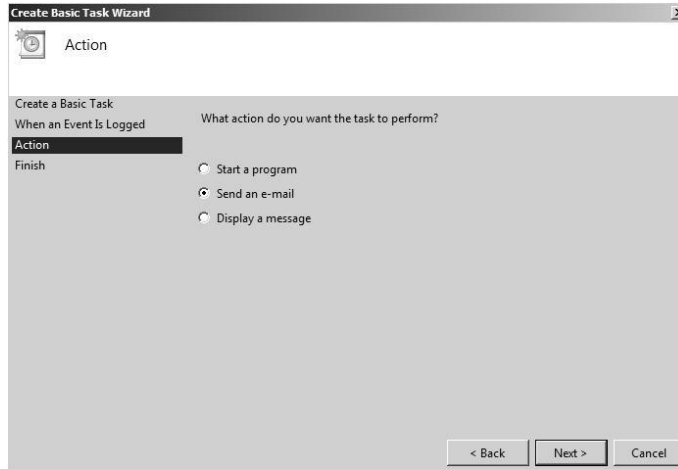


Figure 1-31: Managing and associating an event using an action

TCP/IP network errors

Because Oracle GoldenGate uses TCP/IP network to communicate between source and target systems processes, TCP/IP network related errors response are adjustable to suite the network throughput. Modify TCP/IP values found at `tcperr` file to customizes Oracle GoldenGate TCP/IP related error message, such as time-out interval and dynamic port allocation. The following table shows the default TCP/IP error response settings found at `tcperr` file.

```
#
# TCP/IP error handling parameters
#
# Default error response is abend
#
```

# Error	Response	Delay (csecs)	Max Retries
ECONNABORTED	RETRY	1000	10
#ECONNREFUSED	ABEND	0	0
ECONNREFUSED	RETRY	1000	12
ECONNRESET	RETRY	500	10
ENETDOWN	RETRY	3000	50
ENETRESET	RETRY	1000	10
ENOBUFS	RETRY	100	60
ENOTCONN	RETRY	100	10
EPIPE	RETRY	500	10
ESHUTDOWN	RETRY	1000	10
ETIMEDOUT	RETRY	1000	10
NODYNPORTS	RETRY	100	10

Carefully modify `tcpcorr`, and test using the same network throughput before deploying on production. As unexpected behavior may result due to incorrect settings of delay and retries values when moving from intranet for testing to internet for production TCP/IP network with firewall settings.

Database design considerations

The majority of enterprise applications require data movement originating from heterogeneous databases, from within and outside of the organization. Because Oracle GoldenGate maintains transaction commit order of the source database, the source database layout influences the level of flexibility when implementing Oracle GoldenGate. Data movement is applications transparent eliminating any type of changes. The source system sustains minimum overhead as Oracle GoldenGate is optimized for change data capture from the transaction logs. However; Oracle GoldenGate implementers need to consider the following:

Partitioned schemas

A partitioned schema is forms data model where the schema is self-contained, mostly found on specialized applications with limited number of related database tables. In such environment Oracle GoldenGate achieves the highest level of flexibility during the implementation and deliver sub-seconds latency for data movement, due to no transformation requirements. Partitioned schema helps eliminates data conflict when deploying Oracle GoldenGate on bidirectional topology for active-active environment. For example, in the case of using Oracle sequence as an artificial primary key value on partitioned schema design, associating the generated sequence number with a unique identifier such as the `INSTANCE_NAME` context attribute value always ensures a unique value across the source and target databases, thus avoiding data conflict on bidirectional replication as illustrated next.

For the source database (S1E2),

```
SQL> conn osm$repapi/oracle@S1E2
Connected.
SQL> CREATE SEQUENCE pol_no_seq;
```

Sequence created.

```
SQL> SELECT CONCAT(pol_no_seq.nextval,
2  CONCAT('-', SYS_CONTEXT('USERENV', 'INSTANCE_NAME')) ) POL_NO
3  FROM dual;
```

POL_NO

7-S1E2

SQL>

For the target database (T1C1),

```
SQL> conn osm$repapi/oracle@T1C1
```

```
Connected.
```

```
SQL> CREATE SEQUENCE pol_no_seq;
```

```
Sequence created.
```

```
SQL> SELECT CONCAT(pol_no_seq.nextval,  
2  CONCAT('-', SYS_CONTEXT('USERENV', 'INSTANCE_NAME')) ) POL_NO  
3  FROM dual;
```

```
POL_NO
```

7-T1C1

SQL>

Data types

Depending on the database type, certain data types are not supported. It always possible to migrate unsupported data types to similar functionality data types. For Oracle database, the list is of unsupported data type is limited to: ANYDATA, ANYDATASET, ANYTYPE, BFILE, BINARY_INTEGER, MLSLABEL, ORDDICOM, PLS_INTEGER, TIMEZONE_ABBR, URITYPE and UROWID.

For MS SQL Server, the list of unsupported data types is limited to: SQL_Variant

Referential integrity constraints

Oracle GoldenGate perform transactional change data capture from the transaction logs, preserving the transaction commit order. Even though the ordering of tables found at the extract parameter file is irrelevant. However; the ordering of tables found at the replicate parameter file requires proper listing. When the target database tables constraints are enabled, the order of tables must consider referential integrity constraints among tables, otherwise the replicate processes ABENDED due to constraint violation. This is a challenging task for complex database schemas definition when replicating the entire database.

Three steps demonstration setup

The objectives of presenting an end-to-end configuration is to demonstrate how Oracle GoldenGate fundamental setup is concise and easy when used for as-is data movement such as homogenous replication, which absolutely true when compared with Oracle Steams configuration. The details of the steps are explained in throughout the book.

This fundamental setup can be used for the purpose of quick demonstration or as a reference for time-to-time GoldenGate users. Figure 1-32 shows the topology and the processes and trail files names used throughout the configuration. The source system is running Oracle Database 11g, while the target system is running Oracle Database 12c. The operating system for source and target systems is Oracle Enterprise Linux 5 update 6.

Refer to chapter 2 for details regarding downloading, installing and database pre-requisite of Oracle GoldenGate.

The components used throughout the configuration of the source and target systems are:

- Source database SID: S1E2
- Primary extract process: ES1E201
- Local trail: ./dirdat/sa
- Secondary extract (data pump) process: EPMP01
- Remote trail: ./dirdat/ta
- Replicate process: RS1E201
- Target database SID: T1C1

The unidirectional setup illustrated is using a data pump extract option in a non-ASM Oracle database environment. The database tables used for the demonstration listed below. These table are identical on both systems since no columns mapping required.

- OSM\$REPAPI.CUSTOMERS
- OSM\$REPAPI.POLICIES

Because of referential integrity enabled on the target system, the CUSTOMER is mapped before the POLICIES table. Failure to do so results on referential integrity violation returning an Oracle database error code ORA-02291 integrity constraint violation. In case the target database is for read-only database such as data warehouse, referential integrity constraints may remains disabled.

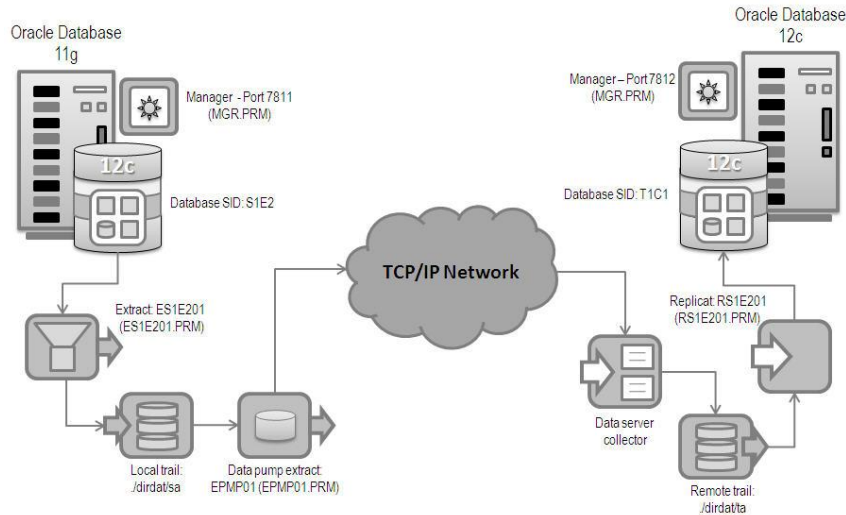


Figure 1-32: Fundamental setup components

Step 1: Primary extract process configuration

The primary extract ES1E201 performs change data capture (CDC) from the transaction logs to keep the source and the target databases synchronized delivering sub-second latency. It's linked with a local trail 'sa' starting at sa000000 and recycling at sa999999. The runtime parameter file is ES1E201.PRM. The primary extract uses in-memory technique to capture currently running transactions, until it's committed and sent to the trail file, or rolled back and discarded by the primary extract. While the primary extract is running, a checkpoint process keep records it read and write position to the checkpoint file. In an event of failure, the primary extract avoid redundant capture of transactions already processed. Oracle GoldenGate uses its own commit sequence number (CSN) to guarantee the transaction integrity and data consistency when applied at the target database by the down-streaming replicate.

-- Parameter file ES1E201.PRM

```
EXTRACT es1e201
SETENV (ORACLE_SID=S1E2)
SETENV (ORACLE_HOME="/u01/app/oracle/product/11.2.0/dbhome_1")
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@S1E2, PASSWORD AACAAAAAAGAIFAAUDVHCFUGFIYF,
ENCRYPTKEY DEFAULT
EXTTRAIL ./dirdat/sa
TABLE osm$repapi.customers;
TABLE osm$repapi.policies;
```

```
GGSCI (ggs-source) 7> ADD EXTRACT es1e201, TRANLOG, BEGIN
```

```
now, THREADS 1
```

```
EXTRACT added.
```

```
GGSCI (ggs-source) 8> ADD EXTTRAIL ./dirdat/sa, EXTRACT
```

```
es1e201, MEGABYTES 32
```

```
EXTTRAIL added.
```

Step 2: Secondary extract process configuration (optional)

The secondary extract process is the data pump, EPMP01. It reads the local trail files then on real-time routes data to the remote data server collector. This data pump uses passthru option because source and target tables structure are identical. The runtime parameters RMTHOST and MGRPORT specify the remote system and Oracle GoldenGate manager listening port number. The remote trail files re-assembled by the data server collector are denoted by RMTTRAIL with the absolute or relative path followed by the name, a two characters prefix for the trail files name starting with ta000000 and recycling at ta999999.

```
-- Parameter file EPMP01.PRM
```

```
EXTRACT epmp01
```

```
USERID ggs_admin@s1e2, PASSWORD AACAAAAAAGAIFAAUDVHCFUGFIYF,
```

```
ENCRYPTKEY DEFAULT
```

```
PASSTHRU
```

```
RMTHOST ggs-target, MGRPORT 7812
```

```
RMTTRAIL ./dirdat/ta
```

```
TABLE osm$repapi.customers;
```

```
TABLE osm$repapi.policies;
```

```
GGSCI (ggs-source) 13> ADD EXTRACT epmp01, EXTTRAILSOURCE
```

```
./dirdat/sa
```

```
EXTRACT added.
```

```
GGSCI (ggs-source) 14> ADD RMTTRAIL ./dirdat/ta, EXTRACT
```

```
epmp01, MEGABYTES 32
```

```
RMTTRAIL added.
```

```
GGSCI (ggs-source) 15> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	STOPPED	EPMP01	00:00:00	00:02:27
EXTRACT	STOPPED	ES1E201	00:00:00	00:08:29

```
GGSCI (ggs-source) 16> start extract *
```

```

Sending START request to MANAGER ...
EXTRACT EPMP01 starting

Sending START request to MANAGER ...
EXTRACT ES1E201 starting

GGSCI (ggs-source) 17> info all

```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	RUNNING	EPMP01	00:00:00	00:02:41
EXTRACT	RUNNING	ES1E201	00:07:55	00:00:02

```

GGSCI (ggs-source) 18>

```

Step 3: Replicate process configuration

The last down-streaming group of the configuration is the replicate process. The replicate process applies transactions to the target database with data consistency and integrity. The replicate relies on the commit sequence number (CSN) for ensuring transactions commit order of the source database. When the target database is solely for the purpose of data warehouse repository to support business intelligence application, then it's required to enable the primary key constraints and indexes, the referential integrity constraints should remain disabled for read-only application. However; when the target database is for load-balancing active-active bidirectional environment, referential integrity constraints must be enabled to ensure data consistency, data integrity and data conflict resolutions.

```
-- Parameter file RS1E201.PRM
```

```

REPLICAT rs1e201
SETENV (ORACLE_SID=T1C1)
SETENV (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
USERID ggs_admin@t1c1, PASSWORD oracle
handlecollisions
ASSUMETARGETDEFS
DISCARDFILE ./dirrpt/rs1e201.dsc, PURGE
MAP osm$repapi.customers, TARGET osm$repapi.customers;
MAP osm$repapi.policies, TARGET osm$repapi.policies;

```

```

GGSCI (ggs-target) 6> ADD REPLICAT rs1e201, EXTTRAIL
./dirdat/ta

GGSCI (ggs-target) 7> info all

```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	STOPPED	RS1E201	00:00:00	00:00:05


```
GGSCI (ggs-target) 9> start replicat *
```

Sending START request to MANAGER ...
REPLICAT RS1T201 starting

```
GGSCI (ggs-target) 10> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RS1E201	00:00:00	00:00:00

Oracle GoldenGate on multitenant container database

Deploying Oracle GoldenGate 12c on an Oracle Database 12c container database (CDB) inherits multitenant database operating requirements for pluggable database (PDB) as source and target databases. Figure 1-33 depicts Oracle GoldenGate for capture from T2C1PDB01 pluggable database.

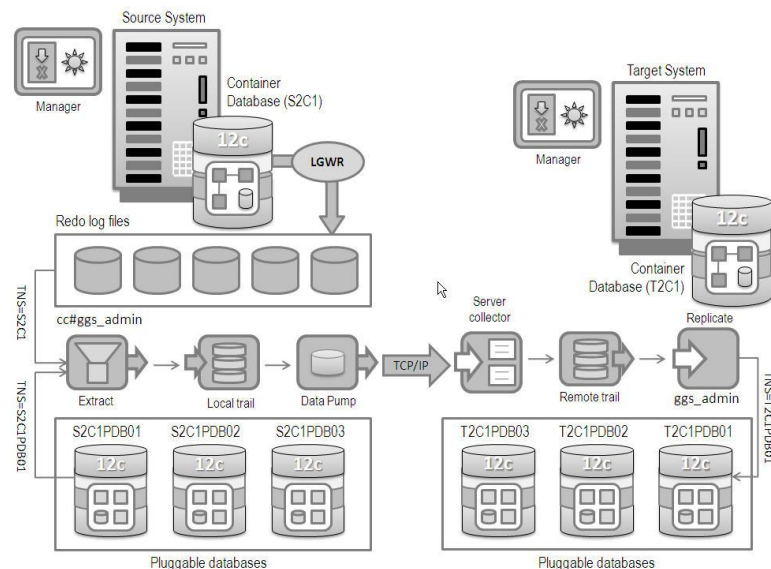


Figure 1-33: Pluggable database Support

Service name

A pluggable database (PDB) has a service name associated with it, the service name matches the pluggable database name. In order to connect from within Oracle GoldenGate to the specific pluggable database, the network configuration is required. The following is portion of the TNSNAMES.ORA for connecting to the T2C1PDB01 pluggable database as Oracle GoldenGate target database.

```
# $ORACLE_HOME/network/admin/tnsnames.ora
#
T1C1PDB01 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (HOST = ggs-target) (PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = T1C1PDB01)
    )
  )
```

Test the connection to the pluggable database before creating the local type database user. The built-in function SYS_CONTEXT returns the session current container database name.

```
$ . oraenv
ORACLE_SID = [T2C1] ?
The Oracle base remains unchanged with value /u01/app/oracle
$ sqlplus /nolog

SQL*Plus: Release 12.1.0.1.0 Production on Mon Sep 16 03:36:59 2013

Copyright (c) 1982, 2013, Oracle. All rights reserved.

SQL> connect system/oracle_4U@t2c1pdb01
Connected.
SQL> SELECT sys_context('USERENV', 'CON_NAME')
       2 FROM dual;

SYS_CONTEXT('USERENV','CON_NAME')
-----
T2C1PDB01

SQL>
```

Source database user

In a multitenant database, the transaction logs (online redo log files) are part of root level database. Oracle GoldenGate capture process must use a common user for change data capture from the transaction logs.

```
SQL> show con_name

CON_NAME
-----
CDB$ROOT
SQL> create user c##ggs_admin identified by oracle;

User created.

SQL> grant dba to c##ggs_admin;

Grant succeeded.

SQL>
```

Target database user

GoldenGate change data delivery process (replicate) requires a local database user for pluggable database (PDB). The user login to the pluggable database to perform database SQL calls to apply changes.

An error is returned when using common user at the root level database.

```
SQL> show con_name

CON_NAME
-----
T2C1PDB01
SQL> create user ggs_admin identified by oracle;

User created.

SQL> grant dba to ggs_admin;

Grant succeeded.
```

The above GRANT command grants DBA privilege to the local user ggs_admin, which is a highly powerful database access rights and should not be the practice on production environment. Refer to chapter 2 to learn more about source and target database privileges required by Oracle GoldenGate extracts and replicate processes.

Checkpoint table

The database schema, ggs_admin, for the checkpoint table specified when adding the replicate process via ADD REPLICAT command or globally on GLOBALS parameter file uses the pluggable database local schema, explicitly connecting to the pluggable database using the network alias T2C1PDB01 from GGSCI command line.

```
GGSCI (ggs-target) 1> dblogin userid ggs_admin@t2c1pdb01, password
oracle
Successfully logged into database T2C1PDB01.
```

```
GGSCI (ggs-target) 2> add checkpointtable
```

```
No checkpoint table specified. Using GLOBALS specification
(ggs_admin.chkpttab)...
Logon catalog name T2C1PDB01 will be used for table specification
T2C1PDB01.ggs_admin.chkpttab.
```

```
Successfully created checkpoint table T2C1PDB01.ggs_admin.chkpttab.
```

```
GGSCI (ggs-target) 3>
```

Change data delivery

The pluggable database is determined by the USERID parameter of the replicate process parameter file. The USERID parameter resolves the pluggable database service name. Now, there are two replicate processes RS1E201 and RPDB01, both use the same remote trail ./dirdat/ta. RS1E201 applies transactions to non-CDB database T1C1, while RPDB01 applies transactions to the pluggable database T2C1PDB01.

```
-- Parameter file: rpdb01.prm
--
```

```
replicat rpdb01
setenv (ORACLE_SID=T2C1)
setenv (NLS_LANG=AMERICAN_AMERICA.AL32UTF8)
userid ggs_admin@t2c1pdb01, password oracle
assumetargetdefs
discardfile ./dirrpt/rpdb01.dsc, purge
map osm$repapi.customers, target osm$repapi.customers;
map osm$repapi.policies, target osm$repapi.policies;
```

```
GGSCI (ggs-target) 3> ADD REPLICAT rpdb01, EXTTRAIL ./dirdat/ta
REPLICAT added.
```

```
GGSCI (ggs-target) 5> START REPLICAT rpdb01
```

```
Sending START request to MANAGER ...  
REPLICAT RPDB01 starting
```

```
GGSCI (ggs-target) 6> INFO ALL
```

Program	Status	Group	Lag at Chkpt	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	RPDB01	00:00:00	00:00:27
REPLICAT	STOPPED	RS1E201	00:00:00	00:13:50

```
GGSCI (ggs-target) 7>
```

Summary

This chapter has covered Oracle GoldenGate fundamentals. The topics presented provided the basis for applying Oracle GoldenGate to an application area requiring design specific configuration to meet the business requirements. Areas where Oracle GoldenGate commonly deployed are:

- Query offloading to reporting instance
- Business continuity for high-availability solutions
- Zero downtime migration and upgrade
- Data warehouse and operational business intelligence
- Data integration services

Oracle GoldenGate topologies and architecture flexibility extends the deployed application area to suite business requirements. For example, query offloading architecture is deployable as one-to-one or many-to-one configuration, depending on the number of data sources and data consolidation requirements. Chapter 4 discusses the technique from a generic prospective and from applying it to Oracle R12 E-Business suite, an important area for the majority of Oracle customers.

Heterogeneity is a key differentiator for Oracle GoldenGate for supporting an enterprise environment running various database management system. The chapter has covered Oracle GoldenGate 12c for Oracle database MS SQL Server, these platforms commonly found at enterprise data processing environment, moving customers data from the workgroup computing servers running MS Windows to EXADATA machine for data consolidation and business intelligence.

The chapter also discussed very specialized application using Oracle GoldenGate 12c adapters, which are necessary components for extending the capabilities to any form of integration services, and this is a major needs for enterprise customers moving data from the mainframe to Linux/UNIX platforms. With Oracle GoldenGate 12c, it's all about being heterogeneous and real-time data movement.

The next chapter will get you started in preparation for deploying Oracle GoldenGate 12c to meet your business requirements. It covers installation and preparation of Oracle database 11g, Oracle Database 12c and Microsoft MS SQL Server for Oracle GoldenGate 12c. Those who need hands-on need to download the virtual machines from <http://www.oracle-ggs.com> to support the workshops' examples.

Good luck, and let's get started with Oracle GoldenGate 12c, *the Oracle bridge*.