



Oracle White Paper
June 2011

Best Practices for Migrating/Upgrading Oracle Database Using Oracle GoldenGate 11g



Introduction

Since concluding an OEM agreement in 1989, Oracle Corporation Japan (“Oracle Japan”) and Fujitsu Limited (“Fujitsu”) have engaged in various joint efforts to provide solutions that deliver safety and security to clients, including system construction, joint verification, and post-installation support.

In November 2006, Oracle Japan established the Oracle GRID Center¹ with Fujitsu and other grid-strategy partners. The Center features cutting-edge technologies intended to strengthen collaborative systems and generate next-generation grid-based business solutions that optimize corporate system infrastructures.

Lending its full support to establish the Oracle GRID Center, Fujitsu participates in joint technical verification efforts with Oracle Japan at the Oracle GRID Center, drawing on its server and storage products.

This white paper discusses procedures for upgrading Oracle Database using Oracle GoldenGate, Oracle’s real-time data integration software, as well as the results of such upgrades.

¹ A wide range of technical reports is available from the Oracle GRID Center.
Oracle GRID Center (<http://www.oracle.com/jp/gridcenter/index.html>)

Overview.....	3
Oracle GoldenGate 11g.....	5
Verification Environment.....	7
System configuration	7
Verification Results	11
Scenario (1): Migrating/upgrading using export/import	11
Scenario (2): Migrating/upgrading using export/import and GoldenGate 11g.....	14
Using Oracle GoldenGate 11g: Results and impact	18
Conclusion.....	20
Appendix.....	21

Overview

This document describes best practices for effectively migrating/upgrading to Oracle Database, together with the results of verification tests. Migrations/upgrades of Oracle Database generally provide the following benefits:

1. Migration to the latest hardware, including CPU multi-core processors and large-capacity memory devices, significantly improves processing performance.
2. Upgrading to the latest version of Oracle Database provides access to functions that reduce operating costs while maximizing the performance of advanced hardware.
3. Using hardware and software for which maintenance services are provided maintains appropriate system lifecycles.

It is important to note that planning a database migration/upgrade entails numerous issues that require close attention. The most common issue is system downtime. Data migrations in Oracle Database generally involve export/import utilities and backup/restoration methods that usually result in downtimes proportional to the volume of data migrated. Since the performance of existing system hardware significantly affects data migration performance, achieving notable performance improvement is difficult, even with system tuning. Accounting for the possibility of extended system downtimes and failback options in the event of problems is an essential part of migration and upgrade planning.

The best practices introduced in this document address these issues. The methods described herein utilize Oracle GoldenGate 11g (hereafter referred to as “GoldenGate 11g”), Oracle’s real-time data integration software. GoldenGate 11g allows fast data replication of databases between different database versions or operating systems. Using GoldenGate 11g for database migrations/upgrades realizes the following benefits:

1. Zero database downtime
2. Mitigate migration/upgrade risks
3. Easy fall back without data loss after database migration

To migrate a database using the GoldenGate 11g data replication function, you must manually copy the entire database to the new system in advance (initial copy). GoldenGate 11g is then used to automatically replicate the data updated after the initial copy (changed data synchronization), eliminating the need to stop the database to copy large volumes of data and minimizing downtimes required for the migration/upgrade (see Fig. 1) to application switchover related downtime.

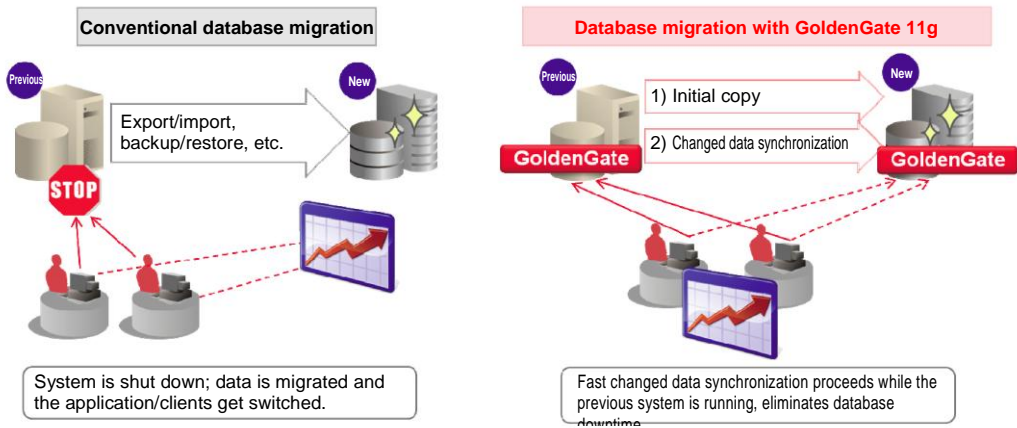


Fig. 1 Comparison of migration/upgrade methods

For example, if you set the day for initial copy and data replication using GoldenGate 11g one month before the migration day, all you have to do on the day of database migration is to stop data replication, check data consistency, and start up the database at the migration destination. In addition to eliminating database downtime, GoldenGate 11g dramatically reduces the work required on the day of migration (system switchover event), mitigates retry risks attributable to human errors, and significantly reduces the man-hours required for repeated processing (see Fig. 2).

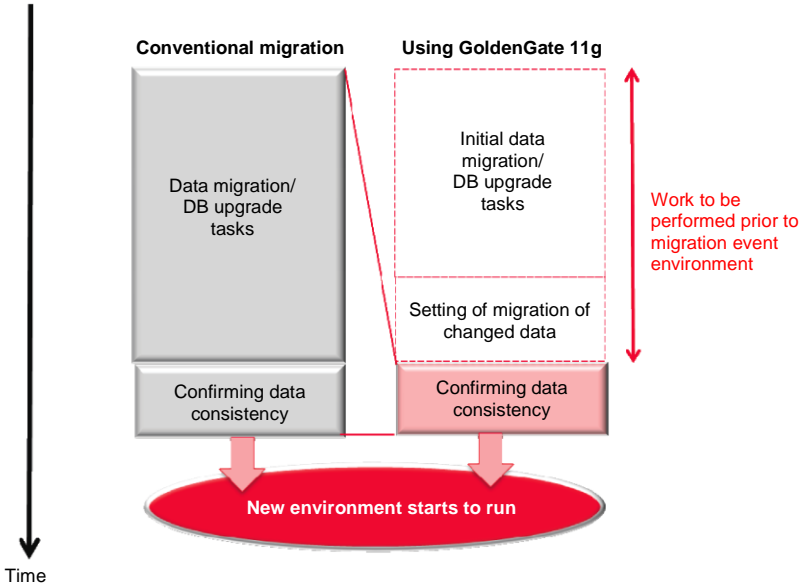
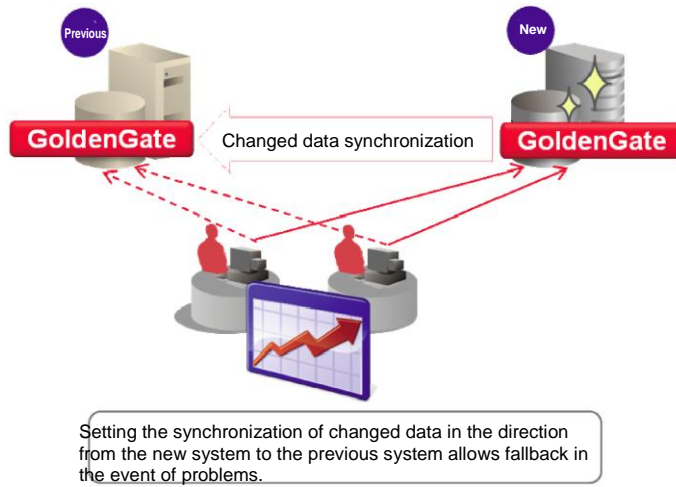


Fig. 2 Comparison of work done on migration day

Replicating data backwards from the new system to the previous system after the database has been migrated reflects post-migration update data in the previous database. This will allow fallback with minimal or no data loss in case of a

post-migration failure (see Fig.



3).

Fig. 3 Fallback in case of failure post-migration

When GoldenGate’s bidirectional replication capabilities are used, both old and new systems can run concurrently. This configuration eliminates downtime completely. Many customers choose this set up to allow phased migration rather than a Big Bang approach. They use both old and new systems in parallel while migrating applications or users in small batches. It allows them to test the new environment completely before retiring the old system.

Fujitsu and Oracle Japan have performed verification testing using an actual system and real-world scenarios to determine the best approaches for minimizing system downtime during migration, one of the GoldenGate 11g benefits described earlier. The following sections will discuss several verification scenarios and the results of using GoldenGate 11g in detail.

Oracle GoldenGate 11g

The first section describes the features and architecture of GoldenGate 11g. GoldenGate 11g is a product designed to replicate data between heterogeneous databases (see Fig. 4).

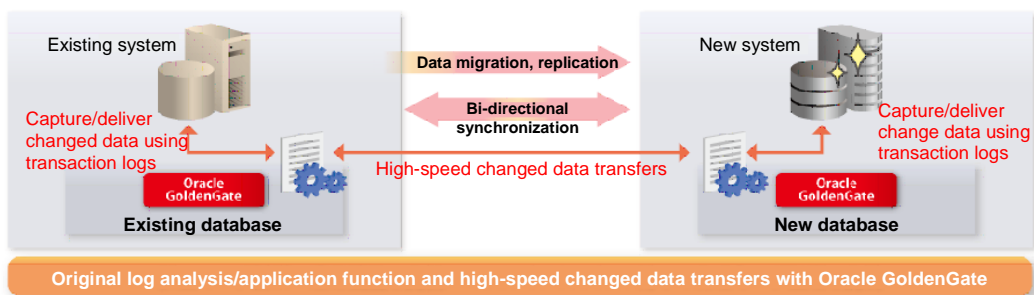


Fig. 4 Overview of GoldenGate 11g

GoldenGate 11g provides the following major features:

Performance

GoldenGate 11g realizes fast, lightweight data replication based on transaction logs (redo logs). It moves only committed transactions and does not impact source or target system performance.

Flexibility

GoldenGate 11g is an open, modular architecture, that supports heterogeneous (different RDBMS or Operating System) replication between multiples sources and targets. It can be used for disaster recovery/data protection, zero downtime migrations and upgrades, operational reporting, real-time BI, query offloading and data distribution.

Reliability

GoldenGate 11g is built upon a highly fault tolerant framework that maintains transactional integrity, that is resilient against interruptions and failures.

Fig. 5 illustrates the architecture of GoldenGate 11g.

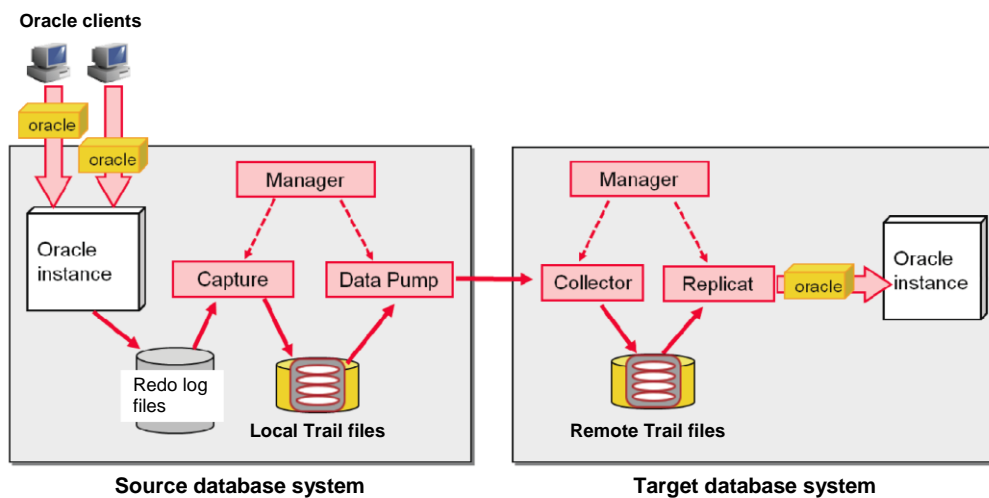


Fig. 5 GoldenGate 11g architecture

Described below are the functions of each component of GoldenGate 11g:

Capture (Extract)

This process extracts database update information from redo logs and writes this data to Trail files, intermediate files in a proprietary format.

Trail Files

Oracle GoldenGate’s unique queuing mechanism—contain the most recent changed data in a transportable, platform-independent format called the Oracle GoldenGate Universal Data Format, and can be converted to XML and other popular formats for consumption by different applications.

Data Pump(Extract)

This process reads update information from Trail files and transfers the data to the target.

Collector

This process writes data sent from Data Pump to Ttrail files.

Delivery (Replicat)

This process converts update information in Ttrail files to SQL statements and sends them to the target database.

Manager

This process monitors and manages processes and Trail files.

Verification Environment

System configuration

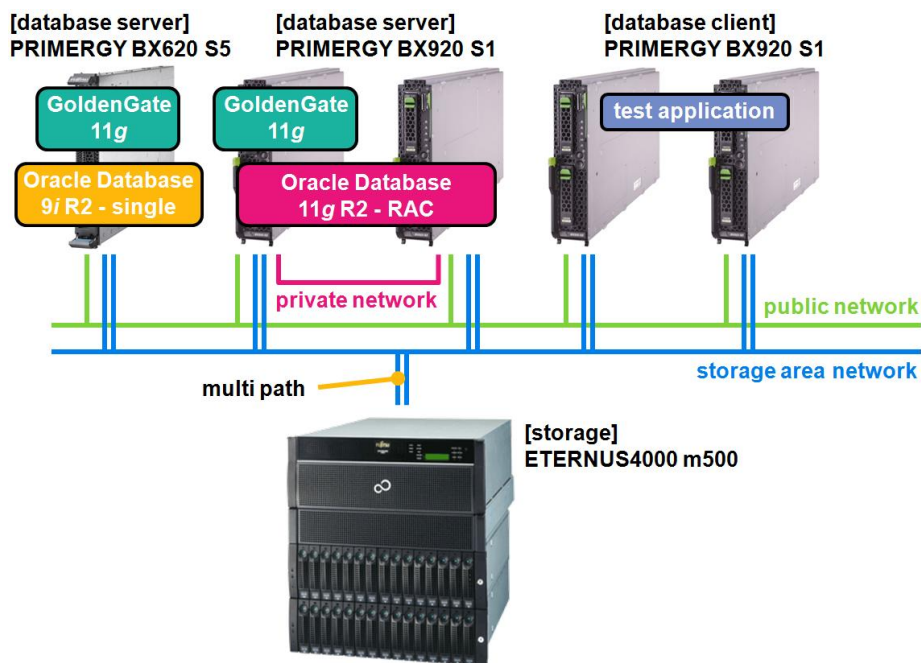


Fig. 6 System configuration

Database server

To create a “previous system” with the source database for our verification tests, we installed Oracle9i Database Release 2 Enterprise Edition in the PRIMERGY BX620 S5 and configured a single-instance database. Then we installed GoldenGate 11g.

MODEL	PRIMERGY BX620 S5
CPU	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz (4core) x 2 Hyper-Threading
MEMORY	24GB

OS	Red Hat Enterprise Linux AS release 4 (Nahant Update 8) x86-64
SOFTWARE	Oracle9i Database Release 2 (9.2.0.8) Enterprise Edition Oracle GoldenGate v11.1.1.0.0 for Oracle 9i on Linux x86-64

To create a “new system” as migration target, we installed Oracle Database 11g Release 2 Enterprise Edition on two PRIMERGY BX920 S1 units and configured two-node Oracle Real Application Clusters. We also installed GoldenGate 11g on one node of the Oracle Real Application Clusters.

MODEL	PRIMERGY BX920 S1
CPU	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz (4core) x 2 Hyper-Threading
MEMORY	36GB
OS	Red Hat Enterprise Linux Server release 5.5 (Tikanga) x86-64
SOFTWARE	Oracle Database 11g Release 2 (11.2.0.2.0) for Linux x86-64 Enterprise Edition Oracle Real Application Clusters Oracle Automatic Storage Management Oracle GoldenGate v11.1.1.0.0 for Oracle 11g on Linux x86-64

Database client

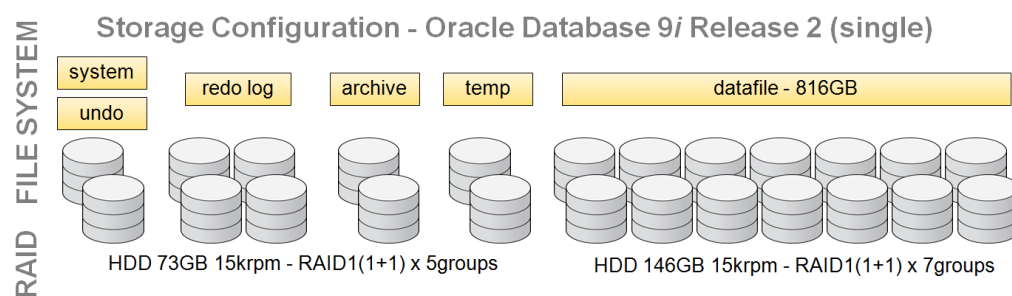
We set up an application that generates OLTP-type loads on two PRIMERGY BX920 S1 units to simulate an online shopping site.

MODEL	PRIMERGY BX920 S1
CPU	Intel(R) Xeon(R) CPU X5570 @ 2.93GHz (4core) x 2 Hyper-Threading
MEMORY	36GB
OS	Red Hat Enterprise Linux Server release 5.5 (Tikanga) x86-64

Storage

MODEL	Fujitsu ETERNUS 4000 M500
DISK DRIVE	HDD 450GB (15krpm) SAS disk drive x 8 HDD 146GB (15krpm) SAS disk drive x 28 HDD 73GB (15krpm) SAS disk drive x 18

Storage configuration



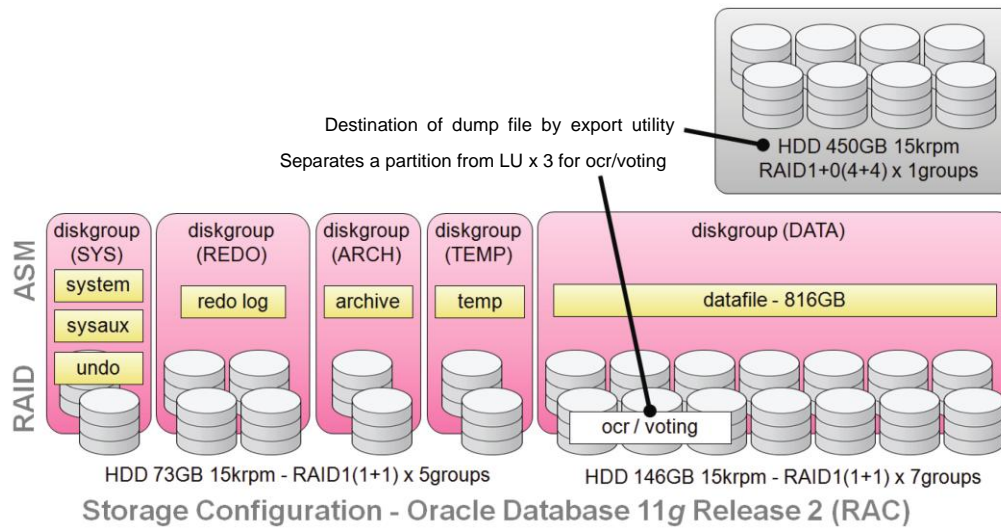


Fig. 7 Storage configuration

Database schema configuration

For our verification tests, we created a schema of 734 GB size using a tablespace/datafile layout as shown in Fig. 7.

Table 4 and

Table 5 below display the schema configuration.

Table 4 Tables

TYPE	TABLE NAME	PARTITION	REMARKS
Master	ACCOUNT	None	User
	PROFILE	None	User profile
	SIGNON	None	Password management
	CATEGORY	None	Product link
	PRODUCT	None	Product master
	INVENTORY	None	Product inventory control
	ITEM	None	Product management
transaction	ORDERS	hash (256)	Order ID and customer name
	ORDERSTATUS	hash (256)	Date of order, and status
	LINEITEM	None	Order quantity and order unit price

Table 5 Indices

TABLE NAME	INDEX NAME	REMARKS
ACCOUNT	PK_ACCOUNT	Index for primary key
PROFILE	PK_PROFILE	Index for primary key
SIGNON	PK_SIGNON	Index for primary key
CATEGORY	PK_CATEGORY	Index for primary key
PRODUCT	PK_PRODUCT	Index for primary key
	PRODUCT_NAME	Product name (for product search)

INVENTORY	PK_INVENTORY	Index for primary key
ITEM	PK_ITEM	Index for primary key
ORDERS	PK_ORDERS	Index for primary key (local)
ORDERSTATUS	PK_ORDERSTATUS	Index for primary key (local)
LINEITEM	PK_LINEITEM	Index for primary key (local)

Verification Results

The following section discusses the effects of using GoldenGate 11g based on comparison of the results of migrating/upgrading Oracle Database obtained in the two scenarios described below.

Scenario (1): Migrating/upgrading using export/import

Scenario (2): Migrating/upgrading using export/import and GoldenGate 11g

Scenario (1): Migrating/upgrading using export/import

Based on this scenario, we verified a method using export/import utility to migrate/upgrade from the previous system configured with Oracle Database 9i Release 2 (9.2.0.8) to a new system configured with Oracle Database 11g Release 2 (see Fig. 8).

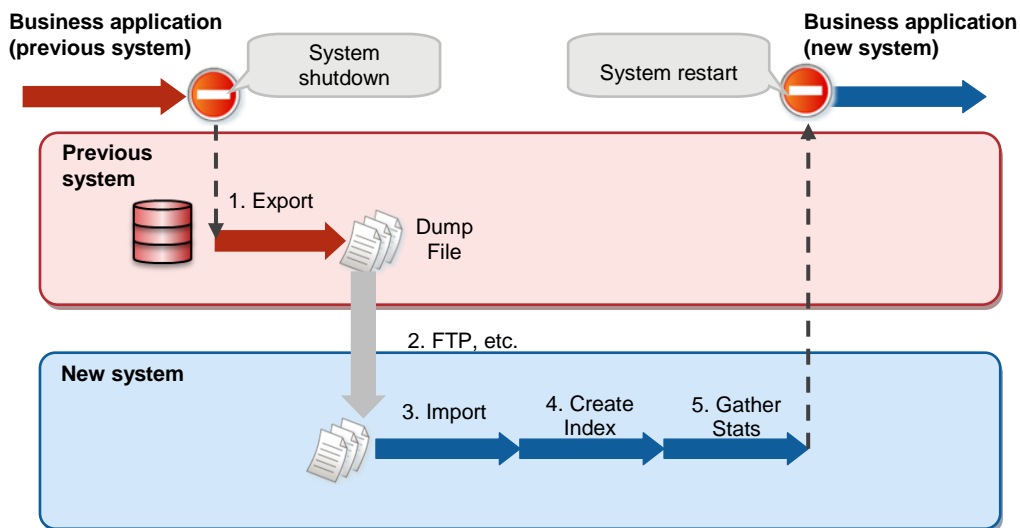


Fig. 8 Migration/upgrade procedure using export/import

The export/import utility is used for migrating/upgrading in cases when it is not possible to upgrade directly from the previous system to the new system due to concurrent hardware migration or differences in platform and/or character sets between the previous and new systems.

If the export is performed while the system continues to operate, specific actions are required to guarantee data consistency during the export. Fig. 9 shows an example of INSERT for Tables A and B from different sessions during the export. In this example, INSERT for Table A is not committed at the start of export and is therefore not written to the dump file. INSERT for Table B, on the other hand, is committed before the start of export and is written to the dump file. If the data in the dump file is imported to the target database, Table A reflects the state before the start of transactions, while Table B reflects the state after the completion of transactions, resulting in data inconsistency.

As described above, data consistency cannot be guaranteed if exports are performed as the system continues to operate. Thus, the system must be shut down before initiating an export, or `CONSISTENT = Y` must be specified.

Specifying `CONSISTENT = Y` allows us to execute an export while maintaining data consistency even if the system is running. It also requires methods to protect against potential export failures (generation of `ORA-01555`) if the undo space required for data consistency is inadequate.

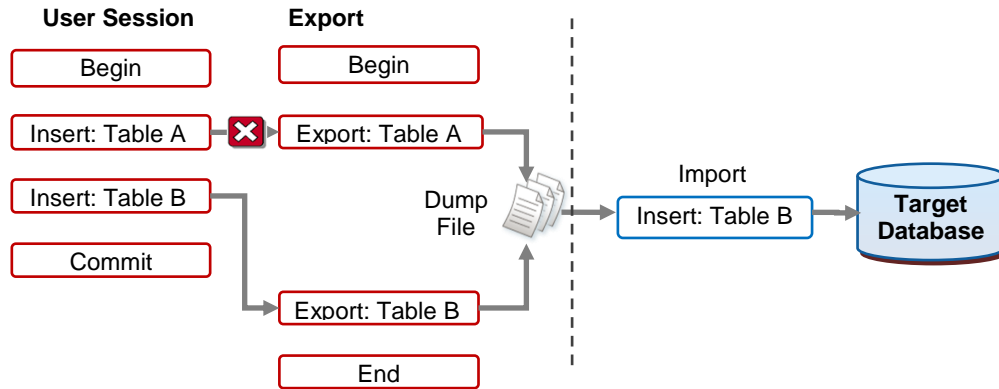


Fig. 9 Data inconsistency problem

Fig. 10 shows the state of business applications under verification. In this verification test, we used an application that generated OLTP-type loads to simulate online shopping transactions as a business application in the source system. We ran the application for roughly one hour. To guarantee data consistency, we stopped² the system during migration work and migrated user data (approximately 580 GB) with the export/import utility. After data migration to the new system completed, we ran the application on the new system. We measured the system downtime defined as the time elapsed from the point at which the application was halted to the time it was restarted, including the time required for each task of the migration/upgrade procedure.

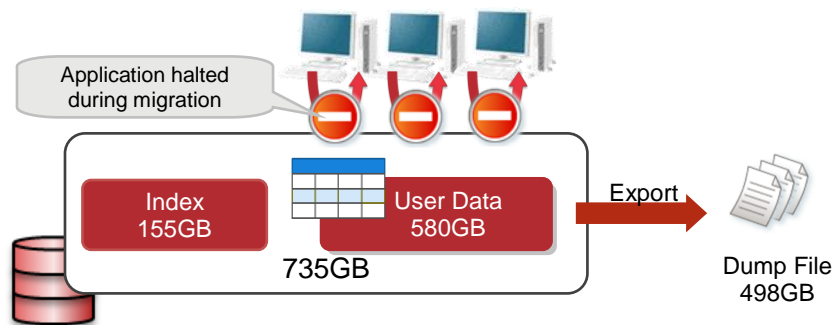


Fig. 10 State of business application during migration

² When we initiated the export simply by setting `CONSISTENT = Y`, the OLTP application and test data used in our verification testing generated large update volumes, resulting in a shortage of undo space and export failure (generation of `ORA-01555`).

Table 6 shows the results of verification testing. According to verification results, the system had to be shut down for roughly 12 hours. The time required for the data export (approx. 3 hours) and import (approx. 6.5 hours) accounted for most of this downtime.

Table 6 Time required to complete the scenario

Step	Time
0.Stop Application	01:00
1.Export	03:30
2.Ftp	01:16
3.Import	06:26
4.Create Index *1	00:07
5.Gather Stats	01:06
Total	12:25
Restart Application	few minutes

(HH:MM)

* The primary key is created at the time of import, and only one other index is created, reducing the time required for processing.

As shown above, migrating/upgrading a database using the conventional export/import utility offers high flexibility, but the system must be shut down for many hours during the migration to guarantee data consistency. Additionally, since system downtime increases in proportion to data volume, a large database can result in significant system downtime—a major issue.

Scenario (2): Migrating/upgrading using export/import and GoldenGate 11g

In scenario (2) (see Fig. 11), we used a combination of the export/import utility and GoldenGate 11g to migrate/upgrade from the previous system configured with Oracle9i Database Release 2 to a new system configured with Oracle Database 11g Release 2.

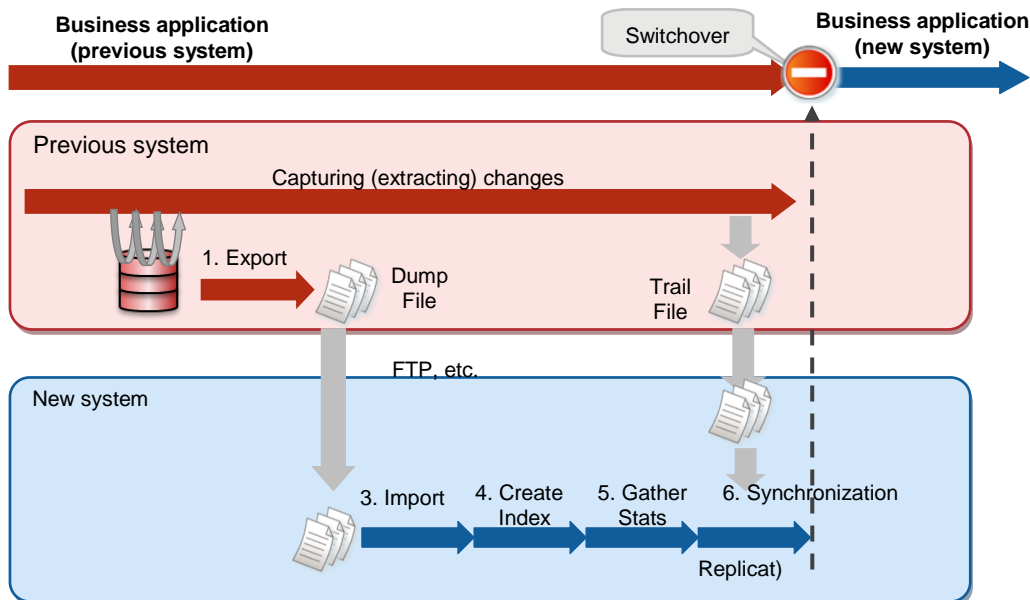


Fig. 11 Migration/upgrade procedure using export/import and GoldenGate 11g

In the verification of the migration/upgrade using the conventional export/import utility based on scenario (1), we must shut down the system for many hours to ensure data consistency. In scenario (2), we can minimize system downtime by using both GoldenGate 11g and the import/export utility.

GoldenGate 11g acquires information on updates at the transaction level executed in the source database from the redo logs and reflects this in the target database in real time. We used this function to apply updates executed in the previous system during the migration process to the database in the new system.

In the example shown in Fig. 12, INSERT for Table A is not written to the dump file, while INSERT for Table B is written to the dump file, as in scenario (1). Scenario (2) differs from scenario (1) in the use of GoldenGate 11g to capture transactions executed in another session during the export. INSERT for Table A is captured by GoldenGate 11g and transferred to the target database. In the target database, GoldenGate 11g reflects INSERT for Table A after the import has completed, setting the database to a state identical to the state at the time of the transaction completion.

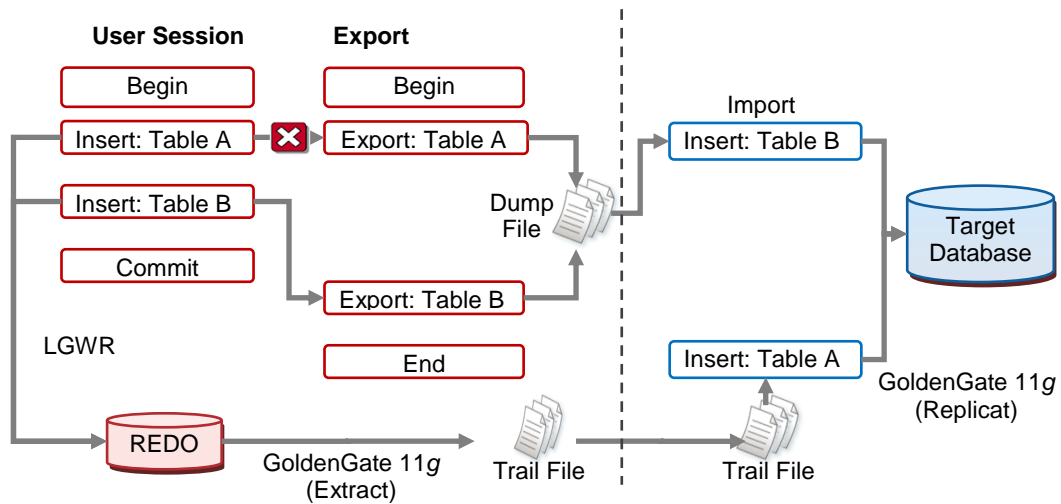


Fig. 12 Resolving data inconsistency problems with GoldenGate 11g

Fig. 13 shows the state of the business application during migration. In scenario (2), the OLTP application runs for one hour as a business application in the source system, just as in scenario (1). Approximately 580 GB³ of user data is then migrated without shutting down the system. GoldenGate 11g captures all new transactions while the migration is done.

After collecting statistical data on the new system has finished, GoldenGate 11g applies and synchronizes transactions executed in the previous system during the migration to the database on the new system. Once the synchronization between the previous system and new system is completed, the system initiates the switchover process to run the application in the new system. We measured the time required by each task of the migration/upgrade procedure and the time required to completely synchronize the previous system and the new system.

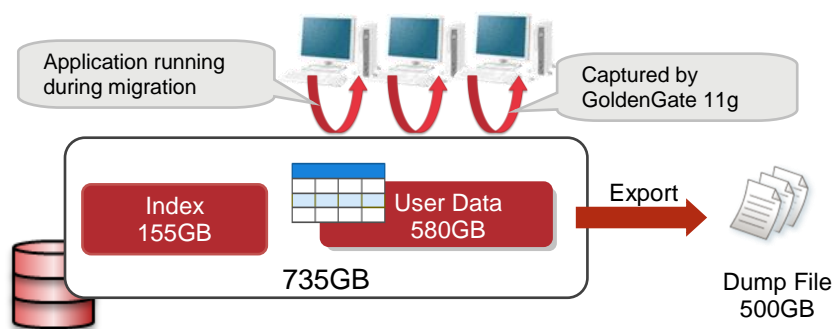


Fig. 13 State of business application during migration

³ In scenario (2), the application runs during the migration/upgrade; thus, the volume of data is slightly greater than in scenario (1).

Table 7 shows the results of the verification test. Roughly 23 hours have elapsed from export start to the completion of the switchover to the new system. However, since the system m.

Table 7 Time required for completing the scenario (comparison)

Step	Time (Scenario 1)	Time (Scenario 2)
0.Stop Application	01:00	01:00 (23:05)
1.Export	<u>03:30</u>	<u>04:39</u>
2.Ftp	01:16	01:16
3.Import	06:26	06:30
4.Create Index *1	00:07	00:07
5.Gather Stats	01:06	01:10
6.1 Synchronization (HANDLECOLLISIONS)		06:02
6.2 Synchronization (NOHANDLECOLLISIONS)		02:21
Total	<u>12:25</u>	23:05 (1~5 13:42)
7.Restart Application (Switch)	few minutes	<u>few minutes</u> (HH:MM)

* The primary key is created at the time of import and only one other index is created, reducing the time required for processing.

Compared below are Step 1 (export) to Step 5 (collection of statistical data), performed in both scenario (1) and scenario (2). The time required for Step 2 through Step 5 was roughly equal for scenario (1) and scenario (2), but the time (04:39) for Step 1 in scenario (2) was about one hour longer than in scenario (1). We suspect that executing export without shutting down the system in scenario (2) increases the amount of HDD read processes performed by both export and business applications, creating a bottleneck in storage I/O performance that extends the processing time (see Fig. 14).

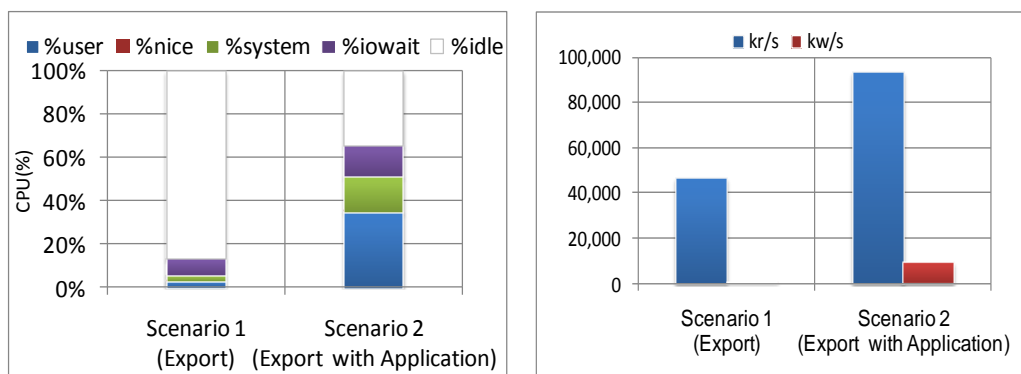


Fig. 14 Comparison of CPU usage and I/O during export

Synchronization by GoldenGate 11g

Next, we examined Step 6 (synchronization by GoldenGate 11g). GoldenGate 11g undertakes two types of synchronization processes: a process in which the Replicat parameter specifies NOHANDLECOLLISIONS (default) and a process in which the Replicat parameter specifies HANDLECOLLISIONS. Ordinarily, if the transactions captured from the source database cannot be applied to the target database, the synchronization process stops to avoid data inconsistencies. However, if HANDLECOLLISIONS is specified, synchronization continues, even if the following errors occur:

- The row to INSERT already exists (table in which primary key/unique key exists).
- The row to UPDATE does not exist.
- The row to DELETE does not exist.

Note: The PK or UK must be added on the target table in order that HANDLECOLLISIONS option works correctly.

Since the transactions that occur from Step 0 to the completion of Step 1 may contain duplicate data, we specified HANDLECOLLISIONS and END RUNTIME first, then executed synchronization. When END RUNTIME is specified, Replicat automatically halts after data updated up to the start of Replicat is applied. Since the transactions captured from the start of Replicat do not contain duplicate data, we specified NOHANDLECOLLISION and then started to execute synchronization.

Fig. 15 compares the number of update records (Capture) obtained per unit time of previous system and the relative value of the number of DML statements (Replicat) processed per unit time in the new system in Step 6. In this scenario, the synchronization process (catchup process) from the previous system to the new system is performed more than three times faster. Using this ratio, we can calculate the time required to synchronize data in the previous system and new system under near-real-time conditions.

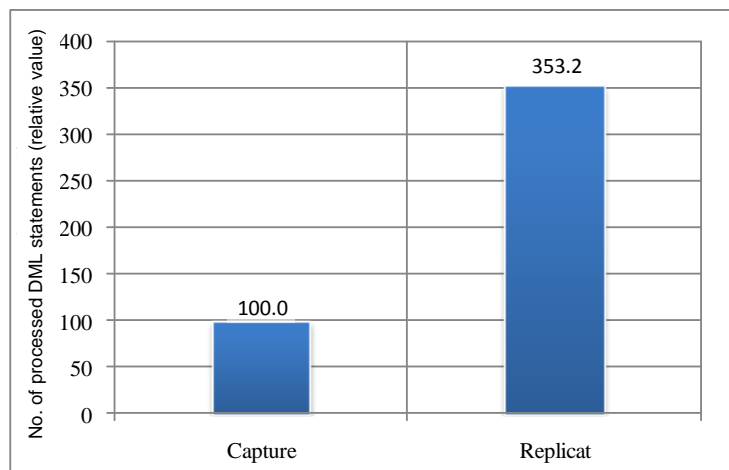


Fig. 15 Number of processed SQL statements

Using Oracle GoldenGate 11g: Results and impact

Fig. 16 compares system downtimes on the day of migration in scenario (1) (migration/upgrade using conventional export/import utility) and scenario (2) (migration/upgrade using export/import utility and GoldenGate 11g).

For the method based on the conventional export/import utility alone, system shutdown and data migration are performed on the day of migration. The system cannot be restarted for more than 12 hours. For the method based on the combination of the export/import utility and GoldenGate 11g, initial copying and changed data synchronization can be performed before the day of the migration without shutting down the system. This minimizes system downtime to just few minutes on the day of migration.

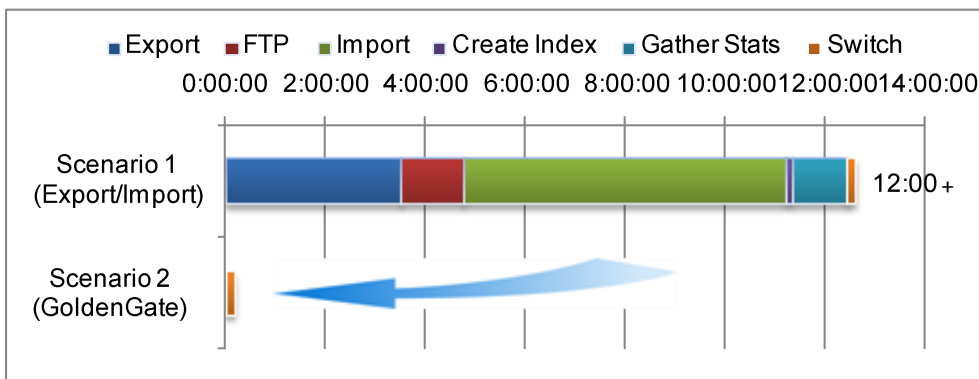


Fig. 16 Comparison of system downtimes on the day of migration

Next, we examine the impact of GoldenGate 11g. To understand how the deployment of GoldenGate 11g impacts the OLTP application, we compared the relative values of throughput and response time. This comparison confirmed that the application used in this verification test achieved almost the same performance regardless of the use of GoldenGate 11g.

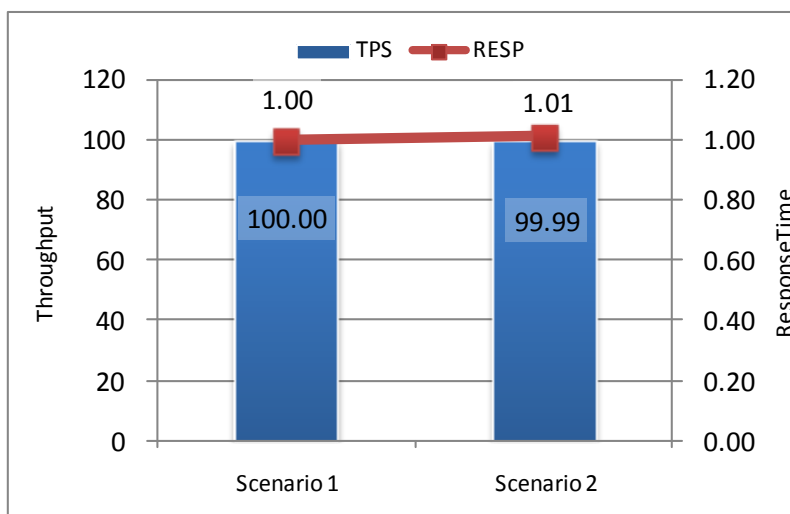


Fig. 17 Impact of GoldenGate 11g on the application

Fig. 18 shows a comparison of impact on CPU utilization. The graph shows that there is virtually no impact on the CPU utilization. We suspect that the slight increase in *%iowait* resulted from the operation to read data from redo logs.

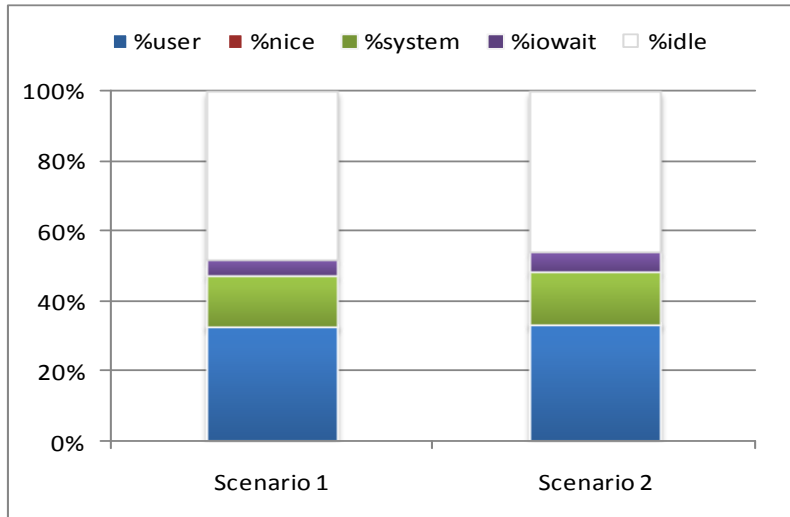


Fig. 18 Comparison of CPU utilization

Fig. 19 compares the impact on redo logs based on the relative I/O values. GoldenGate 11g obtains update information from redo logs, generating read operations to read data from redo logs. Furthermore, supplemental logging must be enabled to identify targets based on the column used for unique identification of changed rows (such as primary key). This results in a moderate increase in writes per second.

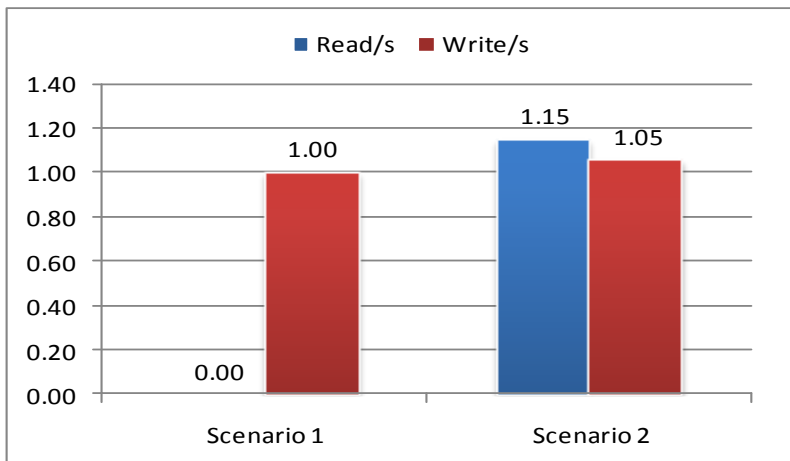


Fig. 19 Comparing I/O to and from redo logs

Conclusion

Our verification tests demonstrate clearly: GoldenGate 11g eliminates database downtime when migrating/upgrading Oracle Database. The benefit of using GoldenGate 11g in such a scenario is not limited to the export/import method of instantiating the target database, it will give the same excellent results when using online backups or other methods for the initial target database setup. In addition, Oracle GoldenGate provides failback option by keeping the old system up to date with the new one, in case there are issues in the new production environment.

The scale and function of a system determines the permissible system downtime. In general, larger systems of higher criticality tolerate shorter system downtimes. Clients may wish to migrate their database but cannot shut it down due to the nature of the system; clients may wish to minimize system downtime to avoid losing business opportunities; or clients may simply wish to minimize the impact on productivity. These are key issues for companies. GoldenGate 11g helps addressing these system migration issues, minimizing system downtime, risks and impact on business activity resulting from database migrations/upgrades.

Appendix

The commands used in our verification tests are given below.

Scenario (1)

Export of data

```
$ exp system/oracle owner="(JPETSTORE)" recordlength=65535
filesize=34359738360 file=exp1.dmp,exp2.dmp,exp3.dmp,exp4.dmp,exp5.dmp,
exp6.dmp,exp7.dmp,exp8.dmp,exp9.dmp,exp10.dmp,exp11.dmp,exp12.dmp,exp13.dmp,
exp14.dmp,exp15.dmp,exp16.dmp,exp17.dmp,exp18.dmp,exp19.dmp,exp20.dmp,exp21.
dmp,exp22.dmp,exp23.dmp,exp24.dmp,exp25.dmp,exp26.dmp log=exp.log
statistics=none compress=n direct=y
```

Import of data

```
$ imp system/oracle filesize=34359738360 file=exp1.dmp,
exp2.dmp,exp3.dmp,exp4.dmp,exp5.dmp,exp6.dmp,exp7.dmp,exp8.dmp,exp9.dmp,
exp10.dmp,exp11.dmp,exp12.dmp,exp13.dmp,exp14.dmp,exp15.dmp,exp16.dmp,exp17.
dmp,exp18.dmp,exp19.dmp,exp20.dmp,exp21.dmp,exp22.dmp,exp23.dmp,exp24.dmp,
exp25.dmp,exp26.dmp fromuser='(JPETSTORE)' touser='(JPETSTORE)'
recordlength=65535 buffer=33554432 statistics=none log=imp.log indexes=n
```

Creating indices

```
SQL> create index product_name on product(lower(name)) parallel 16
nologging;
```

Collecting statistical information

```
SQL> exec dbms_stats.gather_schema_stats(ownname=>sys_context('USERENV',
'CURRENT_USER'),degree=>16,estimate_percent=>1)
```

Scenario (2)

GoldenGate Extract parameter file

```
EXTRACT nex
USERID ggowner, PASSWORD ggowner
EXTTRAIL ./dirdat/nl
TABLE jpetstore.*;
```

GoldenGate Data Pump parameter file

```
EXTRACT ndp
PASSTHRU
RMTHOST f21-64, MGRPORT 7809
RMTTRAIL ./dirdat/nr
TABLE jpetstore.*;
```

GoldenGate Replicat (NOHANDLECOLLISION) parameter file

```
REPLICAT nrp
USERID ggowner, PASSWORD ggowner
ASSUMETARGETDEFS
DBOPTIONS SUPPRESSTRIGGERS
DISCARDFILE ./dirrpt/nrp.dsc, PURGE
MAP jpetstore.*, TARGET jpetstore.*;
```

GoldenGate Replicat (HANDLECOLLISION) parameter file

```
REPLICAT nrp
USERID ggowner, PASSWORD ggowner
ASSUMETARGETDEFS
DBOPTIONS SUPPRESSTRIGGERS
HANDLECOLLISIONS
END RUNTIME
DISCARDFILE ./dirrpt/nrp.dsc, PURGE
MAP jpetstore.*, TARGET jpetstore.*;
```

Process startup commands

```
• GoldenGate Extract startup
GGSCI> start extract nex

• GoldenGate Data Pump startup
GGSCI> start extract ndp

• GoldenGate Replicat startup
GGSCI> start replicat nrp
```

Best Practices for
Migrating/Upgrading Oracle Database
Using Oracle GoldenGate 11g
June 2011



Fujitsu Limited
Shiodome City Center
1-5-2 Higashi-Shimbashi, Minato-ku,
Tokyo, Japan

This document is intended to provide technical information on the results of verification tests performed at the Oracle GRID CENTER. The contents of this document are subject to change without notice to permit improvements. Fujitsu Limited makes no warranty regarding the contents of this document; nor does it assume any liability for damages resulting from or related to the contents of the document.

UNIX is a registered trademark of The Open Group in the United States and in other countries. All SPARC trademarks are used under license from SPARC International, Inc. in the United States and in other countries and are registered trademarks of that company in the United States and in other countries. Products bearing the SPARC trademark are based on an architecture developed by Sun Microsystems, Inc. SPARC64 is used under license from SPARC International, Inc. in the United States, and is a registered trademark of that company.

Sun, Sun Microsystems, the Sun logo, Solaris, and all Solaris-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries and are used under license from that company.

Other product names mentioned in this document are the product names, trademarks, or registered trademarks of their respective companies.

Note that system names or product names in this document may not be accompanied by trademark notices (®, ™).



Oracle is committed to developing practices and products that help protect the environment



Authors: Tomohiro Iwamoto, Yosuke
Goto
Oracle Corporation Japan
Oracle Aoyama Center
2-5-8 Kita-Aoyama, Minato-ku,
Tokyo, Japan

This document is provided for informational purposes only. The contents of the document are subject to change without notice. Oracle Corporation Japan does not warrant that this document is error-free, nor does it provide any other warranties or conditions, whether express or implied, including implied warranties or conditions concerning merchantability or fitness for a particular purpose. Oracle Corporation Japan specifically disclaims any liability with respect to this document. This document does not form any contractual obligations, either directly or indirectly. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without prior written permission from Oracle Corporation Japan.

Oracle and Java are registered trademarks of Oracle Corporation and its subsidiaries and affiliates in the United States and in other countries. The company names and product names mentioned may be the trademarks or registered trademarks of their respective companies.