

An Oracle White Paper
February 2010

Zero-Downtime Database Upgrades Using Oracle GoldenGate

Executive Overview	1
Goals.....	2
Concepts and Terminology.....	2
Source and Target.....	2
Oracle GoldenGate.....	2
Database Upgrade Options	3
Database Migration	3
Standby Database	4
Transportable Tablespace	4
Cross-Platform Transportable Tablespace.....	4
Clone Database.....	4
Oracle Recovery Manager.....	5
Upgrade and Migration Challenges	5
Revenue Impact	5
Customer Expectations and Loyalty.....	5
Interdependencies and Business Reputation.....	6
Instantiation	6
Incremental Data Movement.....	7
Performance Impact	8
Staging Areas.....	8
Change Management	8
Special Handling of Legacy Data or Certain Datatypes.....	8
Verification.....	8
Failback.....	8
Achieving Zero-Downtime with Oracle GoldenGate	8
Overview of Oracle GoldenGate Architecture	9
Capture	10
Trail Files.....	10
Delivery	10
Oracle GoldenGate Veridata.....	11

Detailed Steps Using a Platform Migration Example	11
Migration Without Failback	11
Migration with Failback	13
Failback Steps	14
Partial Migration Using Bidirectional Synchronization	14
Conclusion	15

Executive Overview

Eliminating database downtime poses a significant challenge for IT organizations that need to upgrade or migrate mission-critical database environments running Oracle Database 8*i*, Oracle Database 9*i*, or Oracle Database 10*g* to Oracle Database 11*g*. This is particularly true for applications that must provide continuous or near-continuous operations to users who increasingly expect uninterrupted availability of online services. Any outage of an application or website—even if that outage is scheduled or planned—has an impact on the revenue and reputation of the business.

For databases that host the data store for these mission-critical applications, availability requirements have become stringent. Unfortunately, there are essential events that require application downtime, including modifying hardware or database software, upgrading applications, applying software patches, and migrating to different computing architectures. Because such events are not conceived via system or data failures, they are aptly classified as planned outages. Empirical data from nontrivial applications deployed in very large database environments demonstrates that minimizing downtime to handle planned outages is a complex, time-consuming, error-prone, and costly process.

This white paper explains how organizations can upgrade or migrate from Oracle Database 8*i*, Oracle Database 9*i*, or Oracle Database 10*g* to Oracle Database 11*g*—without downtime. Using Oracle GoldenGate's real-time data integration and replication capabilities, businesses can perform rolling upgrades, create a clone database to offload instantiation and conversions, keep transactions in sync across the databases, manage partial or phased migrations and upgrades, conduct postupgrade/postmigration data verification, and implement a reliable failback strategy.

Goals

The primary goal of this white paper is to increase awareness of a solution that eliminates database downtime during a planned outage for database upgrades and migrations. As such, resource issues such as disk space and software licensing costs are not discussed—not because such issues are unimportant, but because the need to minimize downtime or completely avoid it outweighs other considerations in demanding, high-availability environments. Due to its scope, this white paper is primarily for advanced database users.

Furthermore, this document is not intended to repeat the upgrade steps, typical warnings, and preparatory details associated with an Oracle Database upgrade. There are many articles and notes that exist on the Oracle Support customer portal, in various books, and on the Web, which outline the operational procedures associated with upgrades and migrations.

The Oracle GoldenGate solution presented here leverages real-time synchronization; clone databases; rolling upgrades; in some cases, transportable and cross-platform transportable tablespaces; and failback. Every effort is made to avoid overhead at the primary database to ensure application availability.

A secondary goal of this white paper is to explain how to minimize the wall clock time required to perform the entire upgrade. Experienced users will realize that wall clock time can be on the order of days or weeks, yet the downtime to the application can still be near zero. The key here, as will be explained, is to offload work processing to additional database copies.

Concepts and Terminology

To help comprehend the solutions, key concepts and terms are outlined in the following subsections.

Source and Target

Throughout this document, the production database is referred to as the source and the secondary copy as the target database.

Oracle GoldenGate

Oracle GoldenGate is a real-time change data capture application that provides guaranteed data capture, routing, transformation, and delivery across heterogeneous business systems. The application uses a low-overhead architecture to capture transactions nonintrusively from a source database by reading online transaction logs, transforming the data when needed, and applying those transactions

with guaranteed integrity to a target database in real time.¹ Oracle GoldenGate's processes run continuously—even bidirectionally—and support high-volume, rapidly changing environments, moving thousands of transactions per second with very low impact. The target database is a transactional replica at a logical level, which can be leveraged for multiple applications, including a rolling database upgrade.

Database Upgrade Options

A database upgrade advances the Oracle Database software release number from one version to another. There are two primary ways to perform an upgrade.

- **In-place upgrade.** An in-place upgrade renders the database inaccessible for business applications while the database software is being upgraded. This procedure entails running an upgrade script (such as for u0902000.sql), recompiling invalid PL/SQL, and downtime that is usually not acceptable in most mission-critical environments. (This white paper does not address in-place upgrades.)
- **Rolling upgrade.** The term *rolling upgrade* refers to upgrading different databases or different instances of the same database, such as in an Oracle Real Application Clusters (Oracle RAC) environment, one at a time, without stopping the database. A rolling upgrade consists of the following high-level steps:
 - The application points to the production database running software version VOLD.
 - A secondary logical database copy is constructed running software version VOLD.
 - The secondary database copy is upgraded to the next database version VNEW.
 - The secondary and primary databases are synchronized.
 - The primary database is shut down.
 - The application is pointed to the secondary database.
 - The primary database is kept in the same version VOLD for failback reasons.

Database Migration

Moving an Oracle Database across different operating systems is a common requirement in many computing environments. A migration enables the underlying operating system or hardware platform to be changed. In Oracle, on-disk file formats are not homogeneous across platforms. Under Oracle 10g compatibility, the on-disk structures for platforms that appear in v\$TRANSPORTABLE_PLATFORM are identical, but the endian format could differ.

¹ Noteworthy is that Oracle GoldenGate can be used when the target and source database software come from different vendors. It can also apply changed data in real time to Java Message Service message queues.

Standby Database

A *standby database* is a copy of the main production database that is maintained for high availability or disaster tolerance purposes. The standby database can be physical or logical.

In the physical standby copy, the database is kept in recovery mode. More specifically, the redo logs of the production database are applied to a mounted copy of the production database. There are some hardware and operating system limitations because redo across Oracle platforms is not always compatible.

A logical standby is a copy of the production database that contains the same objects, but doesn't physically match the structure of the production database copy. It is maintained by instantiating a logical equivalent of the production database and replaying the SQL that modifies the production database at the standby site.

Transportable Tablespace

Transportable tablespace is a feature introduced in Oracle Database 8i that allows nonsystem tablespaces to be moved from one database to another by physically grafting the tablespace datafiles into the control files on the target database, and then importing object metadata into the target database's dictionary. Transportable tablespace has three main phases:

- Exporting the metadata (dictionary data for the objects)
- Transferring the datafiles from one database to another
- Importing the metadata and datafiles

Transportable tablespace sets can be created from an Oracle Recovery Manager (RMAN) backup to avoid downtime on the primary database.

Cross-Platform Transportable Tablespace

Cross-platform transportable tablespace is an extension of the transportable tablespace feature that enables tablespaces to be transported across database platforms. This feature can only be used after the database compatibility has been advanced to Oracle Database version 10.0.0.0 or later. The example provided in the white paper uses this feature in conjunction with Oracle GoldenGate to migrate previous Oracle Database versions to Oracle Database 11g Release 2.

Clone Database

A *clone database* is a database constructed using a restored backup of an existing database, recovered to a point in time, and opened.

Oracle Recovery Manager

Oracle RMAN is a database tool that manages the process of making backups and managing the process of restoring and recovering from them. It is also used for the conversion of endian systems during a cross-platform CONVERT.

Upgrade and Migration Challenges

Upgrade and migration projects pose challenges for mitigating business risk and navigating technology issues. Some of the more-common and critical challenges are listed in the subsections that follow. These issues help to highlight why a real-time, low-overhead solution using Oracle GoldenGate is the best way to perform database upgrades and migrations for mission-critical environments.

Revenue Impact

Put simply, downtime equals lost revenue. Many businesses, such as retail, travel, and banking, no longer have the extended downtime windows for planning upgrades and migrations. However, for scalability, performance, and cost-saving reasons, most database environments can benefit by taking advantage of recent technology advancements: low-cost storage, clusterware enhancements, and software improvements, for example. To capture these gains, businesses must find a way to upgrade or migrate while minimizing the high cost of downtime. Figure 1 highlights the average cost of downtime for various businesses.

Industry Segment/ Application	Average Cost per Hour of Downtime
Financial Markets	\$6.45 million
Credit Card Sales	\$2.65 million
Pay-Per-View Television	\$150,000
Home Shopping	\$113,000
Airline Reservations	\$89,000
Teleticket Sales	\$69,000
Shipping	\$28,000

Source: International Data Corporation and Sun Microsystems
(<http://www.sun.com/datacenter/continuity/availability/>)

Figure 1. Average cost of downtime for business applications (in U.S. dollars)

Customer Expectations and Loyalty

In this internet era, businesses must continue to support online processing beyond conventional business hours. Users expect continuous access to services, and unfortunately for providers, the cost of switching to the competition is negligible. For example, look at what happened when eBay took a 22-

hour outage several years ago on a Thursday. In a combined study by Nielsen Media Research and NetRatings, which measures website usage, it was found that during eBay's downtime, the average time spent per person at Yahoo's auction site the next day increased from 7 minutes to 18 minutes and the number of people using Yahoo's site skyrocketed from 62,000 on Thursday to 135,000 on Saturday. Financial losses to eBay were estimated at US\$3 million to US\$5 million.²

Interdependencies and Business Reputation

E-commerce has resulted in digital business partnerships where nonavailability of critical data at one site might also result in loss of business services at multiple sites—via data access from either Web services, direct querying, or preconfigured batch loading. Extended downtime is becoming difficult to plan for because significant coordination is required with business partners. This results in postponement of upgrades and migrations, which has associated indirect costs, such as running the existing software with complex workarounds for bugs and not being able to take advantage of new software functionality in later releases.

Instantiation

The first technical problem that needs to be addressed when performing a rolling upgrade is choosing the method by which the target database is instantiated. In other words, how do you create a first version of the target database?

The complexity of the problem is magnified when the target database is on a different platform, because physical backups cannot simply be restored at the target and converted into a clone.

To handle multiterabyte databases, procedures such as export/import are tedious and labor intensive. Failures during the instantiation or global data consistency are not adequately addressed with methods that are time consuming, unless some form of isolation is used. Extracting data from the production database over a period of time results in a performance impact on the source database that is likely unacceptable. Therefore, a good instantiation solution must properly address

- Global data consistency
- Efficiency (speed)
- Performance (low impact on the production database)
- Manageability

² Chet Dembeck, "Yahoo Cashes In On eBay's Outage," *E-Commerce Times*, June 18, 1999, www.ecommercetimes.com/perl/story/545.html.

Incremental Data Movement

Following instantiation, businesses must determine the point that demarcates the last committed transaction picked up by the instantiation process, with the subsequent transactions submitted against the production database and the method by which the subsequent transactions get propagated to the target database. Depending on the chosen instantiation method, one or both of these problems could be pertinent.

Figure 2 highlights this challenge.

- Tables MASTER and SLAVE have a parent-child relationship via a foreign key constraint.
- The instantiation method (for example, export, SQL copy) picks up a read consistent version of table MASTER as of system change number (SCN) 2312360.
- The instantiation method picks up a read consistent version of table SLAVE as of SCN 2312777.
- Transaction TM inserts new row R1 into table MASTER at SCN 2412890.

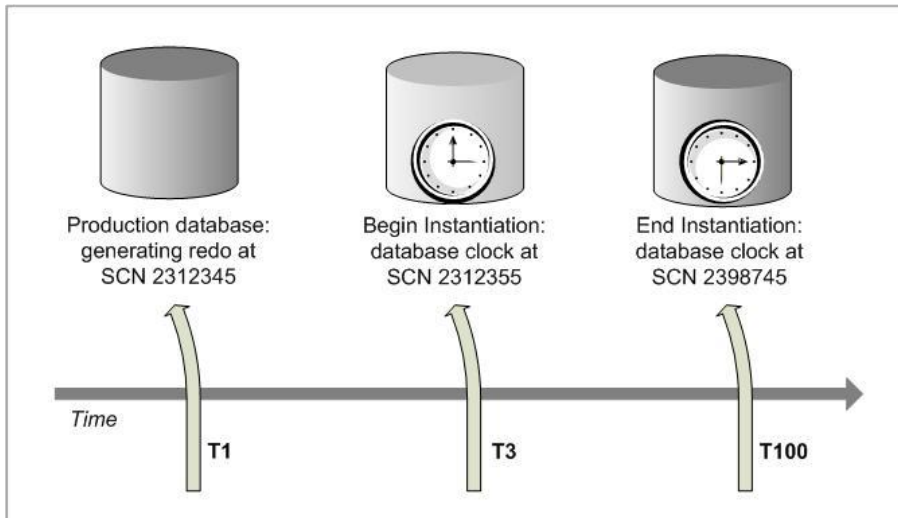


Figure 2. A high-level instantiation of a new target database for upgrades or migrations

There are several problems with this method. First, how are transactions handled that are active as of SCN 2312360 but commit after SCN 2398745? Second, the MASTER and SLAVE tables are from different points in time and, as a result, they are inconsistent with each other from a referential integrity point of view. Third, the data submitted subsequent to SCN 2312360 for the MASTER table (in this case row, R1) still needs to be moved.

A good solution that addresses incremental data movement must

- Ensure global consistency
- Identify a clean instantiation termination point
- Allow transactions from that point onward to be propagated to the target database
- Handle any failures during incremental data propagation (such as loss of network and media failure)

Performance Impact

Another major concern during the upgrade is understanding what kind of impact is experienced by the application running on the production database. The upgrade steps should not degrade performance on the production database such that the application service levels (service-level agreements) are compromised.

Staging Areas

Adequate disk resources need to be configured and managed when addressing rolling upgrades or migrations. Especially when a no-downtime migration is required, proper consideration must be given to handling the disk requirements for the clone database.

Change Management

Changes, other than those made by data manipulation language (DML), need to be addressed during the upgrade or migration process. Examples include datafile additions and PL/SQL package creations. In general, if the overall solution is fast, the changes can be restricted because they are infrequent operations.

Special Handling of Legacy Data or Certain Datatypes

Any datatypes that are not supported by the solution must be identified and handled in advance of the upgrade or migration. Refer to the Oracle documentation for the exact details on supported datatypes for the technologies chosen to perform the zero-downtime migration.

Verification

A postupgrade or postmigration confirmation that the source and target database are synchronized must be conducted. Any out-of-sync conditions at the new source or failback database could pose risks to the business; therefore, they should be identified and resolved. Conducting this verification should be possible even as ongoing transactions are being processed at the source database. Thus, a good solution must address fast and efficient comparison of data across the two databases.

Failback

Once the upgrade or migration is completed, a failback strategy should be in place to avoid any downtime and data loss, in case the new environment is not stable. Therefore, all transactions that have been processed at the target (the new production database) need to be propagated back to the original production database in a consistent and manageable manner for a successful failback.

Achieving Zero-Downtime with Oracle GoldenGate

In-place upgrades require application downtime, which is not feasible in mission-critical, high-availability environments. Therefore, the remainder of this document focuses only on rolling upgrades.

Using Oracle GoldenGate in conjunction with Oracle Database features, a rolling upgrade or a rolling migration can be performed without any application downtime—other than the very minimal time required for application switchover, typically less than one minute and in most cases, only seconds. Using its real-time, heterogeneous data movement technology, Oracle GoldenGate imposes negligible database downtime for upgrades or migrations from Oracle Database 8*i*, Oracle Database 9*i*, or Oracle Database 10*g* to Oracle Database 11*g* Release 2.

If Oracle GoldenGate is set up for bidirectional replication between old and new environments and both systems support the application in transaction processing, end users have the option to implement phased migration to the new environment and eliminates application switchover related downtime. The transition to the new database environment can be completely transparent to the end users.

Upgrading to Oracle 11*g* Release 2 using Oracle GoldenGate consists of the following high-level steps:

1. Set up a standby database running the previous database software version using an existing database backup.
2. Upgrade the standby database to Oracle Database 11*g* Release 2.
3. Synchronize the standby database with the production database.
4. Test in active/live mode.
5. Switch over the application to the standby database.
6. Upgrade the primary database to Oracle Database 11*g* Release 2 after comprehensive application testing at standby.

The next section describes how Oracle GoldenGate can be used to conduct zero-downtime upgrades or migrations. “Detailed Steps Using a Platform Migration Example” discusses the steps in greater detail. Because a migration is a more-complicated procedure than an upgrade, the detailed steps use a case study of migrating an Oracle Database 9*i* on one platform to an Oracle Database 11*g* Release 2 on another platform.

Overview of Oracle GoldenGate Architecture

As shown in Figure 3, Oracle GoldenGate leverages a decoupled architecture comprising independent application modules to capture and replicate data in real time, with low impact on the source database.

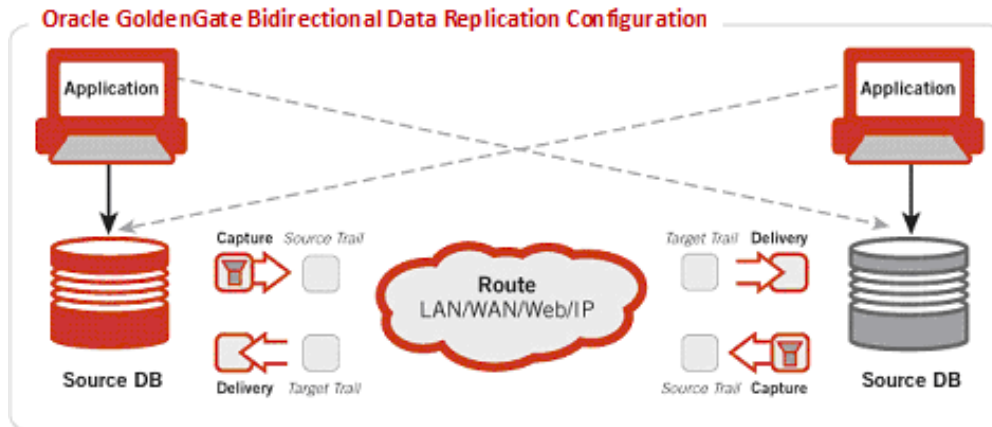


Figure 3. High-level architecture of Oracle GoldenGate, running in a bidirectional configuration

Capture

The Oracle GoldenGate Capture module resides on the source database system and is multithreaded in an Oracle RAC environment. It mines transactions from the Oracle redo log and propagates transactions over the network to an on-disk queue. Only committed transactions are written to the queues.

Initialization

The Capture module can be used either to initialize the target database directly or to start propagating transactions against an existing target database from a given point.

Change Capture

DML and, optionally, data dictionary language (DDL) changes are captured from the transaction logs.

Trail Files

The Oracle GoldenGate Trail Files module can be conceptualized as a persistent ordered set of committed transactions generated by the Oracle GoldenGate Capture process. Trail Files describe DML operations (inserts, updates, and deletes) along with transactional context as captured from the source database.

Delivery

The Oracle GoldenGate Delivery module is a process that runs on the target system. It reads the captured data from the Trail Files module and applies the captured transactions at the target using dynamic SQL. To maintain synchronization between the source and target databases, Oracle GoldenGate applies data changes to the target tables using native database calls, statement caches, and

local database access. To ensure data and referential integrity, Oracle GoldenGate applies data changes in the transaction commit order that occurred at the source database.

Oracle GoldenGate Veridata

Oracle GoldenGate Veridata is a standalone, high-speed data comparison solution that identifies and reports data discrepancies between two databases, without interrupting ongoing business processes. It allows data discrepancies to be isolated for testing and troubleshooting. Oracle GoldenGate Veridata is ideal for conducting data validation after the rolling upgrade, once the source and target are fully operational and running different versions of Oracle Database. It can also help to determine if a failback is needed, in case of any risky data anomalies.

Detailed Steps Using a Platform Migration Example

This is a case study of moving an Oracle Database 9i on IBM AIX to Oracle Database 11g Release 2 on Linux and the detailed implementation steps. These steps can be simplified and adopted to accomplish a rolling upgrade.

Two scenarios are described next: a migration without the addition of a failback solution and the same migration with the addition of a failback.

Migration Without Failback

Figure 4 provides an overview of a database migration without using a failback. The numbers shown in the diagram correspond to the detailed steps below.

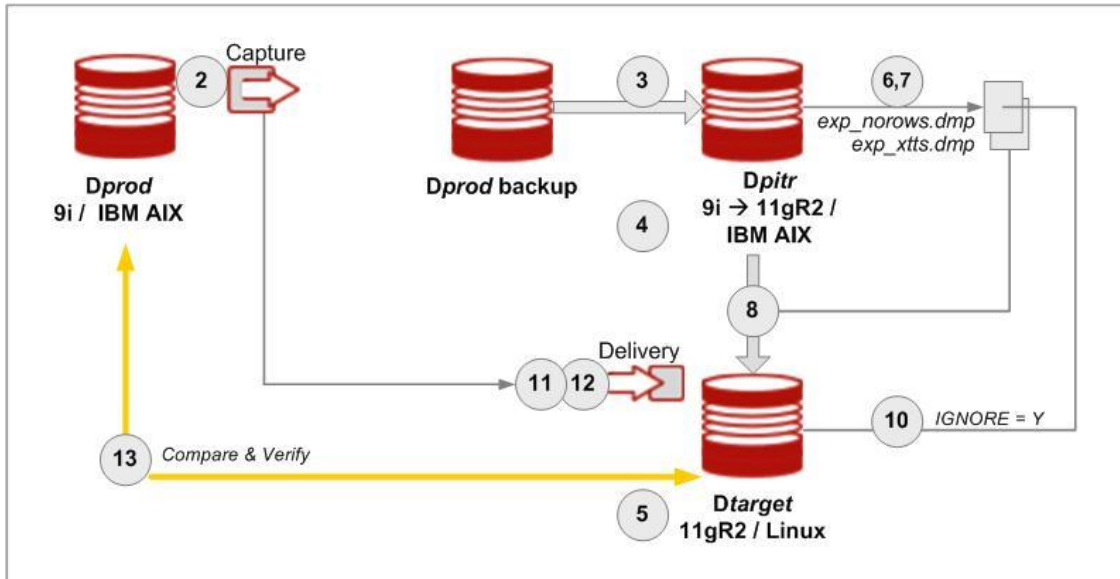


Figure 4. Overview of a cross-platform database migration without failback

1. Address change management by restricting the creation of newer packages and tablespaces during the migration process (not shown). Using DDL migration, these activities can be allowed to some extent.
2. Start the Oracle GoldenGate Capture process at the production database Dprod.
3. Do a point-in-time recovery of an existing backup of Dprod until some SCN Qscn in a separate staging area. Call this database Dpitr. The Capture process in Step 2 must capture all transactions that have a commit time stamp higher than this Qscn. It must have been positioned at or before the beginning of the longest-running transaction when the source database was at Qscn. You can query gv\$transaction and gv\$database at the source database to identify the longest-running transaction and the current SCN at any point in time to identify where Capture should be positioned.
4. Upgrade Dpitr to Oracle Database 11g Release 2 on IBM AIX. Advance compatibility to Oracle Database version 10.0.0.0 or later
5. Set up a vanilla Oracle Database 11g Release 2 database on Linux. Call this database Dtarget.
6. Take a full export without rows at Dpitr. Call the generated export exp_norows.dmp. This will pick up any objects that are not picked up as part of the transportable tablespace export.
7. Unplug the user tablespaces from Dpitr with the Oracle Database cross-platform transportable tablespace feature, using source-side endian conversion. (Note that the conversion would not be required if the endian systems were the same.) This is the step that avoids any performance degradation and does not require any quiescing at Dprod. This step will create a small export dump file. Call this exp_xtts.dmp.
8. Plug the set of tablespaces from Dpitr into Dtarget using the cross-platform transportable tablespace feature. Use the exp_xtts.dmp file created from Step 7. (Note that the plugged in tablespaces are in read-only mode.)
9. Make the set of user tablespaces in Dtarget Read Write (not shown).
10. Do a NOROWS import with IGNORE=Y option at Dtarget using the exp_norows.dmp dump file.
11. Start the Oracle GoldenGate Delivery process at Dtarget and synchronize up to the changes generated since Qscn.
12. If any datatypes are not supported by transportable tablespace or Oracle GoldenGate, then do a special export/import of these objects from Dprod to Dtarget.
13. Use Oracle GoldenGate Veridata to verify that the data at Dprod and Dtarget is synchronized.
14. Switch the application from Dprod to Dtarget (not shown).

The above procedure offloads any quiescing, conversion work to a clone database, and takes advantage of Oracle GoldenGate's incremental real-time data capture and delivery to eliminate the downtime to zero, excluding the application switchover time.

As an alternative to using the cross-platform transportable tablespace feature in Step 6, you can use a full export with data and import into a vanilla database, in which case steps 7, 8, and 9 do not apply. Additionally, in Step 10, you can import the data as well as all database objects.

For upgrading from Oracle Database 10g to Oracle Database 11g, you can use Data Pump Export in parallel to improve performance.

Migration with Failback

Figure 5 provides an overview of a database migration without using a failback. The numbers shown in the diagram correspond to the detailed steps below.

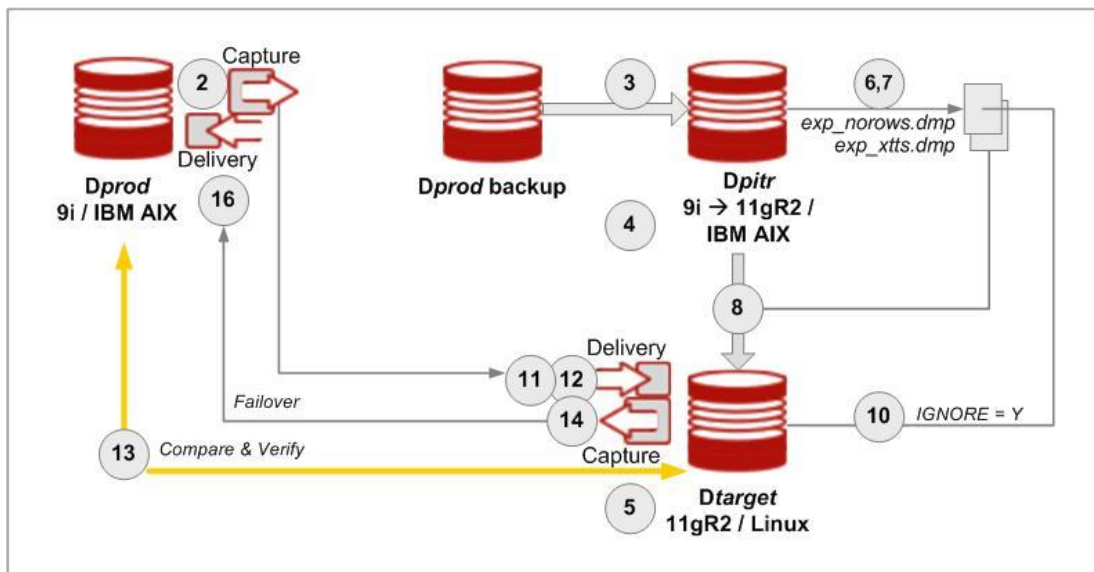


Figure 5. A cross-platform database migration with failback

1. Address change management by restricting the creation of newer packages and tablespaces during the migration process.
2. Start the Oracle GoldenGate Capture process at the production database Dprod.
3. Do a point-in-time recovery of an existing backup of Dprod until some SCN `Qscn` in a separate staging area. Call this database Dpitr. The Capture process in Step 2 must capture all transactions that have a commit time stamp higher than this `Qscn`. It must have been positioned at or before the beginning of the longest-running transaction when the source database was at `Qscn`. You can query `gv$transaction` and `gv$database` at the source database to identify the longest-running transaction and the current SCN at any point in time to identify where the Capture process should be positioned.
4. Upgrade Dpitr to Oracle Database 11g Release 2 on IBM AIX. Advance compatibility to Oracle Database version 10.0.0.0 or later.

5. Set up a vanilla Oracle Database 11g Release 2 database on Linux. Call this database Dtarget.
6. Take a full export without rows at Dpitr. Call the generated export exp_norows.dmp. This will pick up any objects that are not picked up as part of the transportable tablespace export.
7. Unplug the user tablespaces from Dpitr with the Oracle Database cross-platform transportable tablespace feature, using source-side endian conversion. (Note that the conversion would not be required if the endian systems were the same.) This is the step that avoids any performance degradation and does not require any quiescing at Dprod. This step will create a small export dump file. Call this exp_xtts.dmp.
8. Plug the set of tablespaces from Dpitr into Dtarget using the cross-platform transportable tablespace feature. Use the exp_xtts.dmp file created from Step 7. (Note that the plugged in tablespaces are in read-only mode.)
9. Make the set of user tablespaces in Dtarget Read Write.
10. Do a NOROWS import with IGNORE=Y option at Dtarget using the exp_norows.dmp dump file.
11. Start the Oracle GoldenGate Delivery process at Dtarget and synchronize up to the changes generated since Qscn.
12. If any datatypes are not supported by transportable tablespace or Oracle GoldenGate, then do a special export/import of these objects from Dprod to Dtarget.
13. After Oracle GoldenGate eliminates the lag between Dprod and Dtarget, use Oracle GoldenGate Veridata to verify that the data at Dprod and Dtarget is synchronized.
14. Start the Oracle GoldenGate Capture process on Dtarget.
15. Switch the application from Dprod to Dtarget (not shown).
16. Start the Oracle GoldenGate Delivery process on Dprod.

Failback Steps

During the real-time bidirectional data movement, Oracle GoldenGate allows two databases to support transaction processing. This makes failback a trivial process, if the new database is not stable:

1. Stop the application at Dtarget (the new primary) running Oracle Database 11g Release 2.
2. Once Oracle GoldenGate has applied all transactions from Dtarget to Dprod, switch the application to Dprod.
3. Declare Dprod as the new primary.

Partial Migration Using Bidirectional Synchronization

In certain environments, it might be possible to migrate a limited number of applications over to the target database on Oracle Database 11g while running the rest of the applications against the Oracle

Database 8*i*, Oracle Database 9*i*, or Oracle Database 10*g* source database. This can be done based on logical application partitioning or geographical partitioning.

For example, if the application supports 100 bank branches, only one branch might be switched over to the new database until the new Oracle Database 11*g* environment is deemed stable after appropriate testing and validation. Oracle GoldenGate can be run bidirectionally to synchronize the data across the source and the target systems, thus supporting multimaster database environments.

Conclusion

Users expect mission-critical applications to be continuously available. Even planned outages can have a negative impact on user satisfaction. Using Oracle GoldenGate, a rolling upgrade or migration can be completed with zero-database downtime and only very minimal application switchover downtime. Key technical advantages of this solution include

- Rolling upgrades or migrations using two databases
- No instantiation of the primary database by offloading to a clone database
- Conversions offloaded to a clone staging database
- Transaction synchronization across databases
- Data replication and transactional integrity verification using Oracle GoldenGate Veridata
- Database failover using a bidirectional Oracle GoldenGate configuration

Oracle GoldenGate enables the world's largest enterprises to improve the availability, performance, and accessibility of the transactional data that drives mission-critical business processes. Oracle GoldenGate's wide variety of use cases includes real-time business intelligence, query offloading, zero-downtime upgrades and migrations, disaster recovery, and active-active databases for data distribution, data synchronization, and high availability.



Zero- Downtime Database Upgrades Using
Oracle GoldenGate
February 2010
Author: Alok Pareek
Contributing Author: Mark van De Wiel

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com



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

Copyright © 2007, 2010, Oracle and/or its affiliates. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd. 0110