Technical Report

Upgrading to Oracle Database 11*g* with NetApp SnapMirror, FlexClone, and Oracle Real Application Testing

Oracle Alliance Engineering Team, NetApp

December 2009 | TR-3803

EXECUTIVE SUMMARY

This document describes how to upgrade older versions of Oracle® Database running on NetApp® storage to Oracle Database 11*g*. This solution incorporates NetApp SnapMirror® and FlexClone® technology with Oracle Real Application Testing to simulate a realistic workload of a production database system in a test environment before performing the upgrade. This document aims to provide a solution that is cost-effective and helps faster technology adoption at a lower risk.

TABLE OF CONTENTS

1 INTRODUCTION

Today, enterprises have to make sizable investments in hardware and software to roll out any infrastructure changes. For example, suppose an organization considers upgrading its databases to Oracle Database 11*g* to leverage its enhanced functionality and improve database performance. This would traditionally require that the enterprise invest in duplicate hardware for the entire application stack, including Web server, application server, and database, to test their production applications.

Organizations therefore find it very expensive to evaluate and implement changes to their data center infrastructure. Despite extensive testing, unexpected problems are frequently encountered during implementation in the production system. This is because test workloads are typically simulated and are not accurate or complete representations of true production workloads. Data center managers are therefore reluctant to adopt new technologies and adapt their businesses to the rapidly changing competitive pressures.

This solution provides a quick and cost-effective method of upgrading to Oracle Database 11*g* and the complete testing of the production database load in a dev/test environment.

1.1 AUDIENCE

This guide is for NetApp customers using Oracle Database in their production environment on NetApp storage, database administrators, data center managers, sales engineers (SEs), consulting sales engineers (CSEs), professional services engineers (PSEs), professional services consultants (PSCs), contracted delivery partners (CDPs), and channel partner engineers.

This document assumes familiarity with NetApp storage solutions. It is helpful to have a working knowledge of upgrading prior versions of Oracle Database to Oracle Database 11*g*.

1.2 ASSUMPTIONS

Throughout this document it is assumed that there is one primary Oracle Database server/host known as “prod1” running Oracle Database 10.2.0.4 and a dev/test server or host known as “test1” running Oracle Database 10*g* (10.2.0.4) and Oracle Database 11*g* (11.1.0.7.0). Database binaries are installed in two separate Oracle homes.

Two NetApp storage controllers are used. One storage controller named “storage1” is used for the primary database, and the other named “storage2” is used as a dev/test database host.

1.3 SOLUTION SUMMARY

The following steps outline the procedure used to upgrade the database:

1. Mirror the primary database volumes to the secondary storage using SnapMirror.
2. Mount the mirrored volume from the secondary storage in the test database server where Oracle Database 10*g* binaries and Oracle Database 11*g* binaries are already installed in two different `ORACLE_HOME` locations.
3. Start the database using the Oracle Database 10*g* `ORACLE_HOME` and run the tool to upgrade the database to Oracle Database 11*g*.
4. Once the database is upgraded to Oracle Database 11*g*, create Snapshot™ copies of the database volumes and clone these volumes using NetApp FlexClone.
5. Mount the cloned volume on the dev/test host where the Oracle Database 11*g* database binaries are already installed.
6. Capture the full production workload and move it to the test database server.
7. Process the captured workload and replay the captured workload on the secondary database server.

HIGH-LEVEL TOPOLOGY DIAGRAM

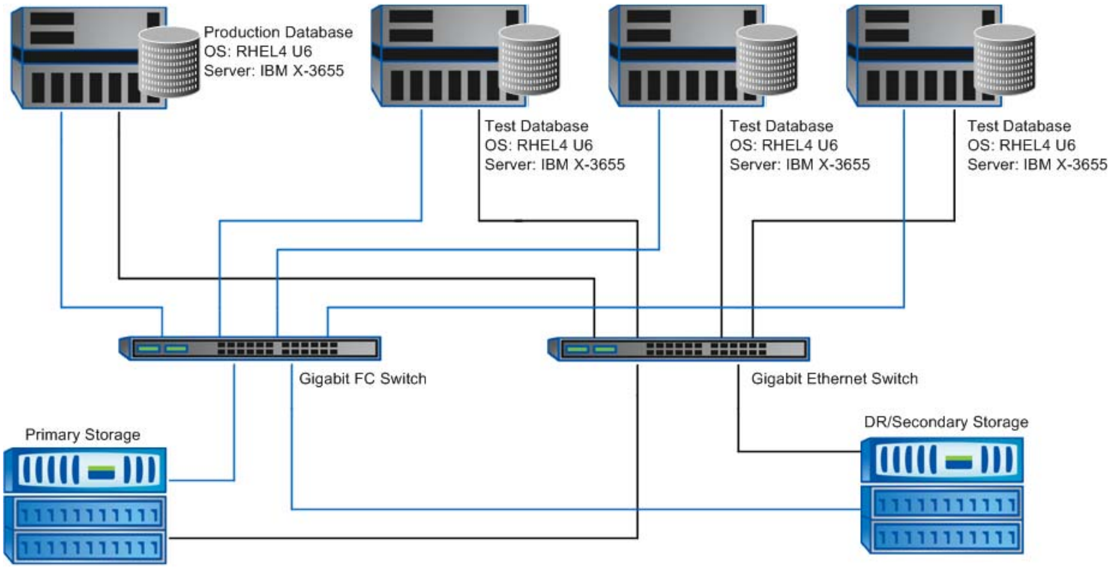

Figure 1) High-level topology diagrams.

LOW-LEVEL TOPOLOGY DIAGRAM

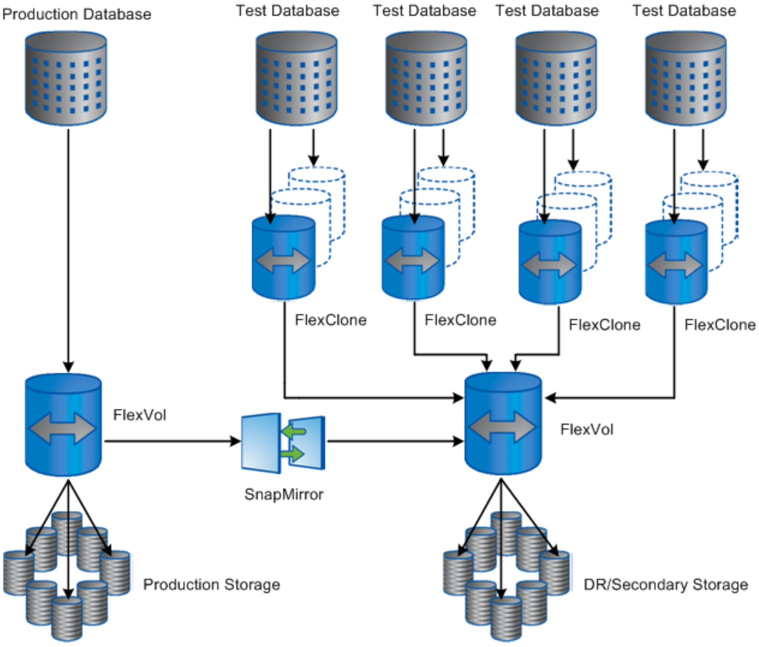

Figure 2) Low-level topology diagrams.

2 NETAPP STORAGE CONFIGURATION

2.1 AGGREGATE AND VOLUME CONFIGURATION

The following tables show the storage layout (aggregate and volumes) for a 1TB Oracle production database used in our setup.

AGGREGATE LAYOUT

Table1) Aggregate layout.

Controller	Aggregate Name	Options	Number of Disks/Type/Size	Purpose
Storage1	ORADATA	RAID-DP® aggr	10 disks/15K RPM FC drives/172GB	Oracle data files, redo logs, archive logs, FRA, and control files
Storage2	ORADATA	RAID-DP aggr	15 disks/10K RPM FC drives/172GB	DR of the primary database

VOLUME LAYOUT

Table 2) Volume layout.

Controller	Volume Name	Aggregate Name	Size	Purpose
Storage1	Oradata_vol	ORADATA	1024GB	Oracle data files and control files copy
Storage1	Redolog_vol	ORADATA	100GB	Redo logs
Storage1	FRA_vol	ORADATA	500GB	Archive logs, FRA, and copy of the control files
Storage2	Oradata_vol	ORADATA	1024GB	Mirror of the Oracle data files and control files from the production storage
Storage2	Redolog_vol	ORADATA	100GB	Mirror of redo logs from the primary storage
Storage2	FRA_vol	ORADATA	500GB	Mirror of the archive logs, FRA, and control files from the primary storage.

Note: The above storage layout is an example used to test the entire solution, which might not be the best recommendation for the Oracle Database. Refer to TR -3411 at http://media.netapp.com/documents/tr-3411.pdf for the best recommendation of storage layout for the database.

2.2 CONFIGURING SNAPMIRROR BETWEEN TWO NETAPP STORAGE SYSTEMS

SnapMirror is a feature of Data ONTAP®, which replicates the Snapshot copy of the data from the source volume or qtree to the destination volume or qtree. SnapMirror operates in either asynchronous or synchronous mode. SnapMirror provides high data availability and fast disaster recovery for your business-critical applications. To learn more about SnapMirror, see reference [7] in Appendix A.

To configure SnapMirror, do as follows:

1. Provide the SnapMirror license on both primary and secondary storage:

   For example:

   ```
   Storage1 > license add <SnapMirror License>
   ```

2. Enable SnapMirror on both storage controllers after adding the license as follows:

   ```
   Storage1 > options  snapmirror.enable  on
   Storage1 > options  snapmirror.access  *
   ```

3. Verify the options set for SnapMirror on both storage controllers as follows:

```
Storage1 > options snapmirror
```

Note: Run the above commands on the secondary storage.

4. To store data mirrored from the primary storage, in the secondary storage, create volumes whose sizes are the same as or larger than the equivalent primary database volumes:

   ```
   Storage2 > vol create <volname>  <aggregate name>  <Size>
   ```

   Example: `Storage2 > vol create oradata  aggr1 50g`

   Note: Asynchronous SnapMirror is used here to mirror the baseline data from the primary (storage1) to the secondary (storage2).

5. Restrict read/write access to the volume created above by entering:

   ```
   Storage2 > vol  restrict  oradata
   ```

6. Initialize the mirror from the primary storage to the secondary storage:

   ```
   Storage2 > snapmirror  initialize  -S   storage1:oradata   oradata
   ```

7. Run the above commands for all other volumes created in the secondary storage to mirror production data.

8. Monitor the progress of the mirror by entering:

   ```
   Storage2> snapmirror  status
   ```

   Mirroring of the volumes is complete when the status column displays “`idle`”.

Note: Run the above commands for all volumes that are used for mirroring the primary database in the secondary storage.

3 INSTALLATION AND CONFIGURATION OF THE DEV/TEST SERVER

Before upgrading the database to Oracle Database 11*g* in the primary database, it is recommended to test the primary load and transaction behavior and performance in a test environment.

To set up and test the server running Oracle Database 10*g*, do as follows:

1 Install the hosts according to vendor-supplied procedures with RedHat Linux® RHEL4 Update 6 with all required patches or rpms for Oracle Database binaries.

2 Before installing Oracle Database binaries, configure kernel parameters and database settings as described in TR-3594 at http://media.netapp.com/documents/tr-3594.pdf.

Note: Refer to TR-3594 at http://media.netapp.com/documents/tr-3594.pdf for details about configuration and Installation of Oracle Database binaries.

4 CLONING THE PRIMARY DATABASE IN A DEV/TEST ENVIRONMENT

After the Oracle Database 10*g* binaries with the required patches (same as the primary database patch set) are installed in the dev/test environment, do the following to clone the primary database:

1. Once the base volume is mirrored sucessfully as described in section 2.2, put the primary database in online/hot backup mode to mirror data from primary storage to the secondary storage as follows:

   ```
   oracle@prod1$ sqlplus “/ as sysdba”
   Sql > alter database begin backup;
   ```

2. Run the following command from the secondary storage to update the mirrored data from primary storage:

   ```
   Storage2> snapmirror update  -S  storage1:oradata  oradata
   ```

3. Monitor the SnapMirror status and release the primary database from the online/hot backup mode once SnapMirror is successfully updated:

   ```
   oracle@prod1$ sqlplus “/ as sysdba”
   Sql > alter database end backup;
   ```

4. Archive the online redo logs on the primary database server:

   ```
   Sql> alter system switch logfile;
   ```

5. Execute the `snapmirror update` command for the archive of the volume once the log switch successfully completes:

```
Storage2 > snapmirror update  -S  storage1:archvol   archvol
```

6. After the update of the volumes is successfully completed, run the following commands to allow read/write access of the secondary storage volumes:

```
Storage2> snapmirror  break  oradata
Storage2> snapmirror  break archvol
```

7. Mount the volumes from the secondary storage on the dev/test host and copy the database parameter file from the primary database server to the dev/test database server. Create all the directory structures (for example, adump, bdump, cdump, udump, and so on) as mentioned in the parameter file on the dev/test server.
8. Start the database in mount mode using the copied database parameter file and then recover the database on the dev/test server:

```
oracle@test1$ sqlplus " /  as sysdba"
sql> startup mount pfile='$ORACLE_HOME/dbs/init<SID>.ora'
sql> recover database using backup controlfile until cancel;
sql> alter database open resetlogs;
```

9. Verify that the database is up and running using the Oracle Database 10*g* binaries.

5 UPGRADING TO ORACLE DATABASE 11*G* IN THE DEV/TEST ENVIRONMENT

After the primary database is cloned on the dev/test server, install Oracle Database 11*g* on the dev/test server using a different `ORACLE_HOME`.

Note: An upgrade from Oracle Database 10*g* to Oracle Database 11*g* in the same server cannot be performed unless both Oracle Database 10*g* and Oracle Database 11*g* binaries exist in the same `ORACLE_HOME`.

To upgrade the database to Oracle Database 11*g*, do as follows:

Step 1

Log in to the system as the owner of the new Oracle Database 11*g* `ORACLE_HOME` and copy the following file from the Oracle Database 11*g* R1 `ORACLE_HOME/rdbms/admin` directory to a directory outside of the Oracle home, such as the `/tmp` directory on your system:

```
oracle@test1$ cp $ORACLE_HOME/rdbms/admin/utlu111i.sql  /tmp/
```

Step 2

Change to the directory where **`utlu111i.sql`** has been copied in step 1.

Start SQL*Plus using the Oracle Database 10*g* `ORACLE_HOME` and connect to the database instance as a user with SYSDBA privileges. Then execute the `utlu111i.sql` file and spool the output:

```
oracle@test1$ sqlplus '/ as sysdba'
SQL> spool upgrade_info.log
SQL>@/tmp/utlu111i.sql
```

Step3

Check the integrity of the source database (Oracle Database 10*g* on the test server) prior to starting the upgrade by downloading and running the **`dbupgdiag.sql`** script listed in Metalink article Note 556610.1.

If the **`dbupgdiag.sql`** script reports any invalid objects, run `$ORACLE_HOME/rdbms/admin/utlrp.sql` multiple times to validate the invalid objects in the database, until there is no change in the number of invalid objects:

```
oracle@test1$ cd   $ORACLE_HOME/rdbms/admin
oracle@test1$ sqlplus "/ as sysdba"
SQL> @utlrp.sql
```

Note: Resolve invalid objects under SYS and SYSTEM prior to the upgrade.

Run the `dbupgdiag.sql` script once more to make sure that no invalid objects appear.

Step 4

If Oracle Database Vault is enabled, disable it before upgrading the database. Enable it again after the upgrade is finished. To do this, see Oracle Metalink Note 453903.1.

Step 5

Stop all database services, including the `listener`, `dbconsole`, and `isqlplus`:

```
oracle@test1$ lsnrctl stop
oracle@test1$ emctl stop dbconsole
oracle@test1$ isqlplusctl stop
```

Also, make sure no data files need media recovery before starting the upgrade:

```
oracle@test1$ sqlplus ‘ / as sysdba’
SQL> SELECT * FROM v$recover_file;
```

The output should not return any rows.

Shut down the database running on Oracle Database 10*g* on the test server:

```
SQL> shut immediate
```

Step 6

Now point all environment variables (for example, `ORACLE_BASE`, `ORACLE_HOME`, `PATH`) to the Oracle Database 11*g* binary directory:

```
oracle@test1$ export ORACLE_BASE=  <Location of Oracle 11g Base>
oracle@test1$ export ORACLE_HOME =<location of oracle 11g home>
oracle@test1$ export PATH= $ORACLE_HOME/bin:$PATH
```

Update the `/etc/oratab` entry to set the new `ORACLE_HOME` pointing toward ORCL and disable automatic startup.

Example: `/etc/oratab`

```
#orcl:/opt/oracle/product/10.2/db_1:N
orcl:/opt/oracle/product/11.1/db_1:N
```

Step 7

At the operating system prompt, change to the `$ORACLE_HOME/rdbms/admin` directory of Oracle Database 11*g* R1 `ORACLE_HOME`. Copy the Oracle Database 10*g* parameter file from the `ORACLE_HOME/dbs` directory to the Oracle Database 11*g* `ORACLE_HOME/dbs` directory and make the necessary changes for Oracle Database 11*g*:

```
oracle@test1$ cd $ORACLE_HOME/rdbms/admin
oracle@test1$ sqlplus "/ as sysdba"
SQL> startup UPGRADE
SQL> set echo on
SQL> SPOOL upgrade.log
SQL> @catupgrd.sql
SQL> spool off
```

Step 8

Run the following postupgrade script to check for invalid objects in the database after completion of the `catupgrd.sql` script:

```
SQL>$ORACLE_HOME/rdbms/admin/utlu111s.sql
```

Run the `utlrp.sql` script to recompile any invalid objects.

Modify the `listener.ora` file to point to the Oracle Database 11*g* home and then start the listener:

```
oracle@test1$ lsnrctl start
```

6 CREATING DATABASE CLONES IN THE SECONDARY STORAGE

After upgrading to Oracle Database 11*g*, create multiple clones of volumes using NetApp FlexClone in the secondary storage. These cloned volumes can be mounted on multiple dev/test database hosts where Oracle Database11*g* binaries are installed, and they will act as independent database instances without occupying extra storage space on the secondary storage.

You can use these independent databases to replay different kinds of workloads captured from the primary database.

To create multiple clones of dev/test databases after upgrading to Oracle Database 11*g* using NetApp FlexClone technology, do the following:

1. Shut down the dev/test database:

   ```
   oracle@test1$ sqlplus "/ as sysdba"
   SQL > shut immediate
   SQL > exit
   oracle@test1$
   ```

2. Create a Snapshot copy of the dev/test database volumes in the secondary storage:

   ```
   Storage2>  snap create <volume name>  <snapshot name>
   Storage2>  snap create  oradata  oradata_snap1
   ```

   Note: Create a Snapshot copy of all the volumes used by the dev/test database.

3. Create clones of the volumes using the Snapshot copy created in step 2:

   ```
   Storage2> vol clone create  <clone name>   -s none   -b  <original volume name>
   <snapshot name>
   Storage2> vol clone create oradata_clone   -s none   -b  oradata  oradata_snap1
   ```

   Note: Create multiple clones of the volumes used for the dev/test database.

4. Mount these cloned volumes on different dev/test hosts where the Oracle Database 11*g* software is installed and start the database on each dev/test host.

   Note: Copy the parameter file from the primary dev/test host to all other dev/test hosts and create all directory structures used for `bdump`, `udump`, `cdump`, `adump`, and so on, on the entire dev/test host before starting the database on each dev/test host.

7 WORKLOAD CAPTURE AND REPLAY ON THE DEV/TEST SERVER

This section describes how to use the new features of Oracle Real Application Testing introduced with Oracle Database 11*g* to simulate the primary database's workloads in a dev/test environment. Historically, the most difficult part of this testing process has been to create something that approaches a realistic workload. Database replay provides the ability to capture an actual workload and "play" it back in the same or a different environment.

7.1 CAPTURING A WORKLOAD

Capturing a workload employs a PL/SQL package interface to configure, start, and stop the capture process. The capture can be filtered to capture only the actions of some of the database users instead of its default behavior to capture all database activity. The capture process operates at a database level, not an instance level, so RAC environments require no special configuration.

The `DBMS_WORKLOAD_CAPTURE` package provides a set of procedures and functions to control the capture process. Before initiating the capture process, an empty directory is required on the primary database server "prod1" to hold the capture logs.

1. Create a directory structure on the primary database server to store the captured logs:

   ```
   oracle@prod1$ mkdir $ORACLE_HOME/cap_dir
   ```

2. Create a directory object on the primary database server "prod1" pointing to the new directory:

   ```
   oracle@prod1$ sqlplus "/ as sysdba"
   SQL>create or replace directory cap_dir as '$ORACLE_HOME/cap_dir';
   ```

3. Make sure the database parameter `pre_11g_enable_capture` on the primary database is set to true before starting the capture of the primary workload:

```
SQL> show parameter pre_11g_enable_capture
```

4. Set the parameter value to true if the above query returns a value of false:

   ```
   SQL>alter system set pre_11g_enable_capture=true scope=both;
   ```

5. Start capturing the primary workload from the primary database server after setting up the above database parameter:

   ```
   SQL>execute DBMS_WORKLOAD_CAPTURE.start_capture (‘name of capture’, ‘cap_dir’);
   ```

6. Continue performing database transactions. Stop the workload capture after one hour of production workload capture:

   ```
   SQL>execute DBMS_WORKLOAD_CAPTURE.finish_capture ();
   ```

7.2 PROCESSING THE CAPTURED WORKLOAD

Once the capture is complete, it must be processed to make it ready for playback. There can be only one set of capture files in a given capture directory. That is because the only parameter given to the DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE procedure is the capture directory where capture files were directed.

Preprocessing does not have to be done on the capture source machine. It is typically done on the same server where the replay is going to take place so that the replay source machine isn't burdened with the additional overhead of preprocessing.

1. Create a directory on the dev/test server to store the captured workload from the primary server. Once the capture of workload is completed, copy the captured files from the primary to the dev/test server:

   ```
   oracle@test1$ mkdir $ORACLE_HOME/rep_cap_dir

   oracle@prod1$ scp -R  $ORACLE_HOME/cap_dir/*
   oracle@test1:$ORACLE_HOME/rep_cap_dir/
   ```

2. Create a directory object in the dev/test database server “test1” pointing to the new directory:

   ```
   oracle@test1$ sqlplus “/ as sysdba”

   SQL>create or replace directory rep_cap_dir as ‘$ORACLE_HOME/rep_cap_dir’;
   ```

3. Process the captured workload using the DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE procedure:

   ```
   SQL>exec DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE (‘rep_cap_dir’);
   ```

7.3 REPLAYING THE CAPTURED WORKLOAD

To replay your processed workload, you must coordinate a few commands—some in the database and some on the operating system. When running database capture and replay, AWR automatically collects Snapshot copies at the start and end of the capture and replay as well as at its normal regular intervals. The replay clients are processes that run on the operating system, outside the database, to execute the workload that was captured and processed.

Another important consideration when replaying the workload is to properly configure the test database that will be used to execute the replay. The target for the replay should have the same data as the primary database did when the capture was started. When these are exactly the same, you can obtain realistic results.

1. Run the following commands to replay the workload in the dev/test database server:

   ```
   oracle@test1$ wrc userid=system password=*******   mode=calibrate
   replaydir=$ORACLE_HOME/rep_cap_dir

   oracle@test1$ sqlplus “/ as sysdba”

   SQL> exec DBMS_WORKLOAD_REPLAY.initialize_replay ( ‘replay name’ , ‘rep_cap_dir’);

   SQL> exec DBMS_WORKLOAD_REPLAY.prepare_replay ();

   SQL > exit;

   oracle@test1$ wrc userid=system password=*******   mode=replay
   replaydir=$ORACLE_HOME/rep_cap_dir
   ```

2. Open a new terminal window in the dev/test database server as the Oracle user and run the following command:

   ```
   oracle@test1$ sqlplus “/ as sysdba”

   SQL > exec DBMS_WORKLOAD_REPLAY.start_replay;
   ```

The replay will complete automatically after some time (depending on captured time).

7.4 ANALYZING REPLAY RESULTS

Once replay is complete, you might want to compare the replay system results with the results from the capture system. The SQL Performance Analyzer, the second feature under the name Real Application Testing, provides an interface to make this comparison.

Using the SQL Performance Analyzer, Workload Report, and AWR Compare Period Report, you can quickly see the effect of your changes—for better or for worse. When attempting to identify how a single change might affect a particular SQL statement, the SQL Performance Analyzer is the best tool. For reviewing overall effect to the whole database, an AWR Compare Period Report will likely be the most insightful.

A report can be generated in text or HTML format using the REPORT function.

For example:

```
oracle@test1$ sqlplus “/ as sysdba”
SQL > SELECT id, name FROM dba_workload_replays;


        ID NAME
---------- ------------------------------
        11 test_capture_1


1 row selected.
SQL>
```

```
DECLARE
  l_report  CLOB;
BEGIN
  l_report := DBMS_WORKLOAD_REPLAY.report(replay_id => 11,
format => DBMS_WORKLOAD_REPLAY.TYPE_HTML);
END;
/
```

Only database replay can conclusively determine that you've tested your database's real workload from your real production system. Combining database replay with flashback database gives you the ability to replay multiple times without having to contemplate a lengthy and laborious restore and recovery process between each replay using SnapMirror and FlexClone.

8 SUMMARY

This document describes an end-to-end solution for upgrading the Oracle Database from an earlier version to Oracle Database 11*g* with real application workload testing. For a problem-free upgrade, this upgrade procedure is first tested in a database environment that is a clone of the production database. A clone of the production is created by using FlexClone.

Please forward any errors, omissions, differences, new discoveries, and comments about this paper to niranjan@netapp.com.

APPENDIX A: REFERENCES

Table3) References.

#	Topic	Reference
1	RAID-DP	http://media.netapp.com/documents/wp_3298.pdf www.usenix.org/publications/library/proceedings/fast04/tech/corbett/corbett.pdf
2	SATA	http://media.netapp.com/documents/rp-0046.pdf http://media.netapp.com/documents/tr-3574.pdf
3	Database installation and configuration	http://media.netapp.com/documents/tr-3594.pdf
4	Database layout	http://media.netapp.com/documents/tr-3411.pdf
5	Snapshot	http://media.netapp.com/documents/tr-3001.pdf http://media.netapp.com/documents/ar1038.pdf
6	FlexClone	http://media.netapp.com/documents/tr-3347.pdf http://media.netapp.com/documents/tr-3348.pdf http://media.netapp.com/documents/tr-3373.pdf http://media.netapp.com/documents/tr-3373.pdf http://media.netapp.com/documents/tr-3646.pdf
7	Thin replication (SnapMirror)	http://media.netapp.com/documents/tr-3446.pdf http://media.netapp.com/documents/tr-3326.pdf www.netapp.com/us/library/research-papers/rp-0007.html

APPENDIX B: UTILITIES

The following table lists the procedures required to start and manage replay in addition to the workload replay client (wrc) utility.

Table 4) Utilities.

Procedure or Utility	Argument	Required or Optional	Description
DBMS_WORKLOAD_REPLAY. INITIALIZE_REPLAY	replay_name	Required	An arbitrary name for the replay.
DBMS_WORKLOAD_REPLAY. INITIALIZE_REPLAY	replay_dir	Required	The database directory object where the capture files are located.
DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY	synchronization	Optional	A Boolean parameter indicating if COMMIT order is preserved from the capture, default TRUE.
DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY	connect_time_scale	Optional	Scaling factor to determine the time elapsed between the capture start and when sessions connected to the database, expressed as a percentage, default 100.
DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY	think_time_scale	Optional	Scaling factor to determine the time elapsed between two successive calls from the same session, expressed as a percentage, default 100.
DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY	think_time_auto_correct	Optional	A Boolean parameter to indicate if replay should adjust think time between calls such that the elapsed replay time matches elapsed capture time, default TRUE.
DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION	connection_id	Required	A connection ID from DBA_WORKLOAD_CONNECTION_MAP.

Procedure or Utility	Argument	Required or Optional	Description
DBMS_WORKLOAD_REPLAY.REMAP_CONNECTION	replay_connection	Required	The new connection string that should be used during replay.
DBMS_WORKLOAD_REPLAY.START_REPLAY	none		
wrc	mode	Required	The replay client mode, one of calibrate, replay, or list_hosts.
wrc	replaydir	Optional	The directory containing replay files; default is the current directory.
wrc	process_per_cpu, threads_per_cpu	Optional	process_per_cpu and threads_per_process are optional parameters (defaults 4 and 50, respectively) that might be provided in calibrate mode.
wrc	userid, password, server, workdir, debug, connection_override	Optional	These parameters might be provided in replay mode. userid and password are self-explanatory. server is the TNS connection string to the replay database. workdir specifies a directory path where debug information might be written if debug is set to files. debug might be set to files, stdout, both, or none (default is none). connection_override is a Boolean when TRUE ignores remapping and uses the value of the server parameter for all connection remapping.

NetApp provides no representations or warranties regarding the accuracy, reliability or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information in this document is distributed AS IS, and the use of this information or the implementation of any recommendations or techniques herein is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. This document and the information contained herein may be used solely in connection with the NetApp products discussed in this document.

NetApp™
www.netapp.com