**UPGRADE / PATCH**

**ORACLE DATABASE**

# Patch Oracle Database

## Check list before patch

* Check java version: 1.5.0 or more
* Copy Opatch and patch to server with **correct owner**
* Create folder .patch\_storage in Opatch

## Method patch

Strong recommend Auto patch

## Normal patch

Example:

---user grid

/u01/app/11.2.0/grid/OPatch/ocm/bin/emocmrsp -no\_banner -output /tmp/file.rsp

---user root

/u01/app/11.2.0/grid/OPatch/opatch auto /u01/setup/23274134/ -ocmrf /tmp/file.rsp

## Post upgrade

# Upgrade Oracle Database

## Check list before upgrade

### Get all the best information!

### Always upgrade Grid (Clusterware and ASM) First!

### Patch your new $ORACLE\_HOME before you upgrade

### Preparation – Testing

### Sanity Operations:

* Invalid Objects

SQL> select unique OBJECT\_NAME, OBJECT\_TYPE,OWNER

from DBA\_OBJECTS

where STATUS='INVALID';

Fix all INVALID objects BEFORE the upgrade/migration

There should be no invalid objects in SYS and SYSTEM user schema

Recompile invalid objects with utlrp.sql before the upgrade

* Always check for DUPLICATE objects in SYS/SYSTEM

SQL> select OBJECT\_NAME, OBJECT\_TYPE

from DBA\_OBJECTS

where OBJECT\_NAME||OBJECT\_TYPE in (select OBJECT\_NAME||OBJECT\_TYPE from DBA\_OBJECTS where OWNER='SYS')

and OWNER='SYSTEM'

and OBJECT\_NAME not in ('AQ$\_SCHEDULES\_PRIMARY', 'AQ$\_SCHEDULES','DBMS\_REPCAT\_AUTH');

Fix DUPLICATE objects in SYS/SYSTEM BEFORE upgrade

Note:1030426.6 Clean Up Duplicate Objects Owned by SYS and SYSTEM

* Invalid Components

SQL> select substr(COMP\_ID, 1,10) compid,

substr(COMP\_NAME,1,24) compname, STATUS,

VERSION from DBA\_REGISTRY where

STATUS<>'VALID';

* Purge RECYCLEBIN

SQL> purge DBA\_RECYCLEBIN;

### Always run the pre-upgrade script:

### Remove "old" parameters, underscores and events from your pfile/spfile

### Speed up your upgrade

* Create Dictionary statistics the night before the downtime window

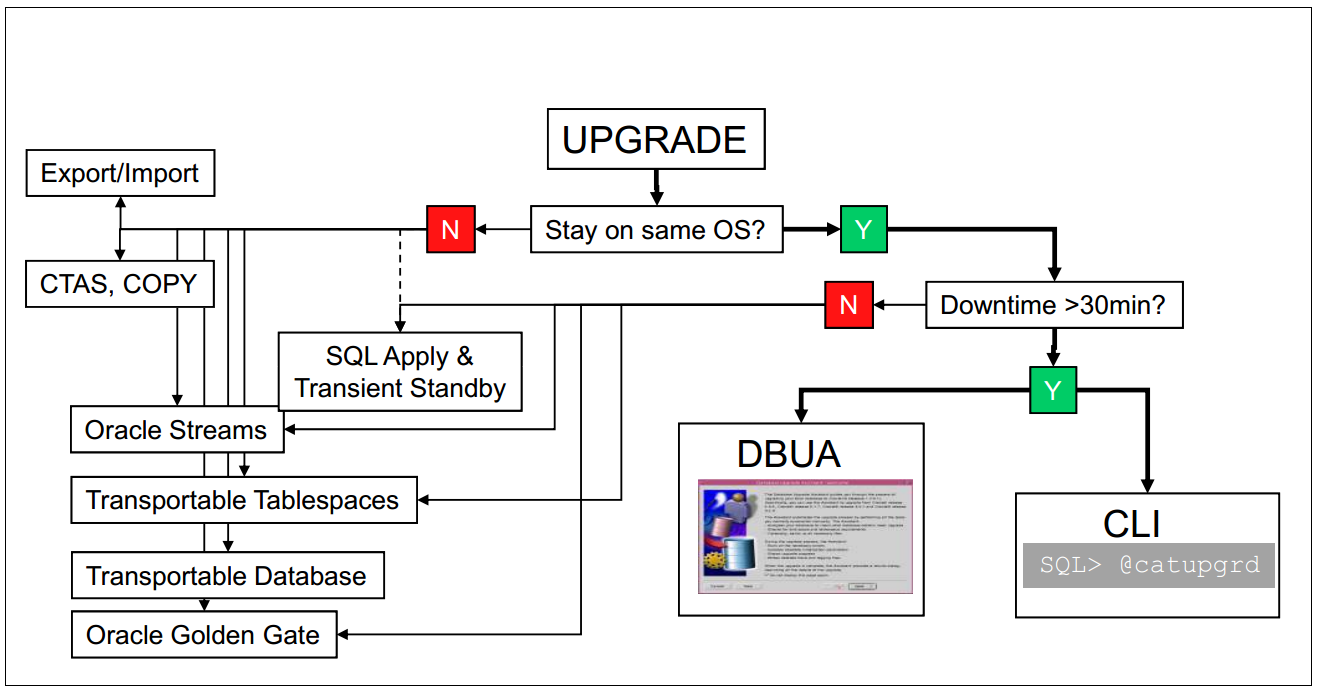
SQL> exec DBMS\_STATS.GATHER\_DICTIONARY\_STATS;

* Truncate or populate AUD$ in SYS/SYSTEM if it has many entries:

SQL> truncate table SYS.AUD$;

## Method upgrade

### Upgrade path



### When to Choose the DBUA

* Can afford 30 – 90 minutes average downtime
* Operating system remains the same
* GUI is preferred over manual command line interface
  + Automatically performs useful pre-upgrade checks
  + Less error-prone / less manual effort
* Note: especially useful for RAC databases
* Consideration:
  + Source and target Oracle Homes must be on the same system
  + Cannot be re-run if an error is encountered mid-upgrade

### When to Choose Command-Line

* Can afford 30-90 minutes average downtime
* Manual command-line interface is preferred over GUI
* Existing database is at least 9.2.0.8
* Migrating to a new hardware platform with same OS

SQL> spool upgrade.log

SQL> @catupgrd.sql

* Consideration
  + Cannot migrate to a different operating system architecture
  + More manual steps required
  + Potential for errors due to typos, missed details

### When to Choose an Alternative Method

* Alternative methods include
  + Original exp/imp or Data Pump expdp/impdp)
  + Oracle Streams or Oracle Golden Gate
  + Data Guard (SQL Apply)
  + Transportable Tablespaces, Transportable Database
  + Moving data via CREATE TABLE AS SELECT or other techniques
* Alternative methods must be used when
  + Moving to a different operating system platform (32- and 64-bit versions of an OS are considered “the same platform” in this case
* Alternative methods may be a good option when
  + Minimal downtime (<30 minutes) required or desired
  + Re-organizing database storage or schemas

## Normal upgrade

### Database Upgrade Assistant (GUI)

Features:

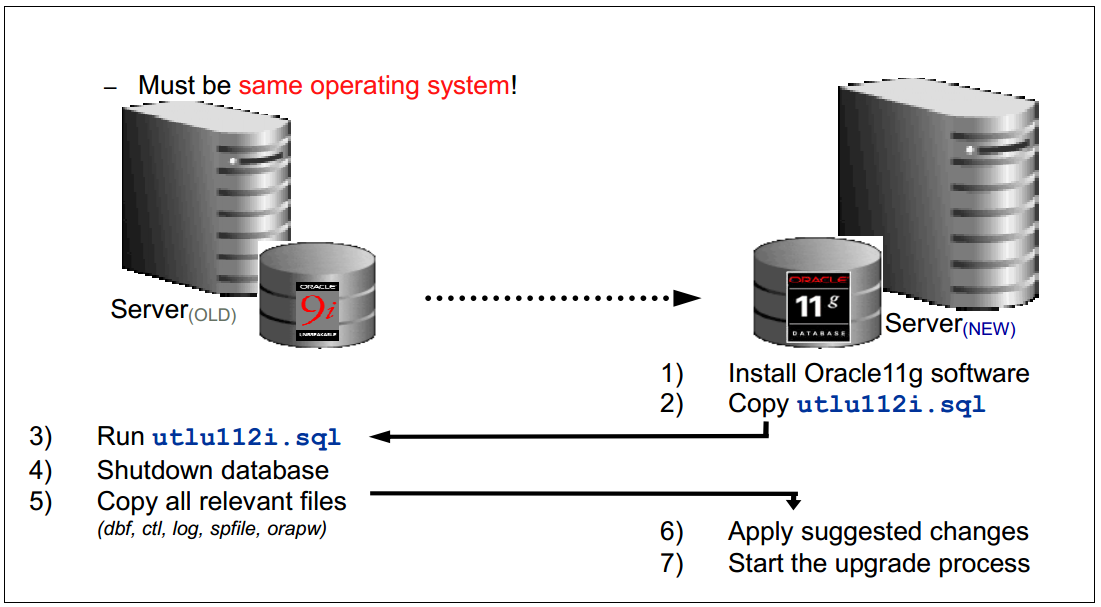
* Graphically led upgrade
* Lots of important checks
* RAC aware - inclusion of all nodes
  + for RAC (almost) a must !!!
* Offline Backup and Restore possible
* ASM upgrade (until 11.1)
* Patchset upgrades
* Logs: $ORACLE\_HOME/cfgtoollogs/dbua

Limitations:

* Only usable if upgrading in place, without moving to new system
* Cannot be restarted if upgrade is interrupted

### Command Line Upgrade

* Typical scenario



* Command Line Upgrade – Step-by-Step
  + Install and patch the new Oracle home
* Take a complete online backup of the current database
* Download and run utlu112i.sql and follow its recommendations
* Copy password file and spfile to the new $OH/dbs
* Switch to the new 11.2 env.
* Start a new 11.2 listener
* Start upgrade

SQL> startup upgrade

SQL> @catupgrd.sql

SQL> @catuppst.sql

SQL> @utlrp.sql

SQL> @utlu112s.sql

SQL> @utluiobj.sql

Time zone change

### Post Upgrade

* SPFILE
  + Always create an editable init.ora from the current SPFILE after the upgrade has been finished
  + Prevents rewrite in case of setting wrong parameters or forced edit
* Time zone:

set serveroutput on;  
declare  
num\_of\_failures number;  
begin  
dbms\_dst.upgrade\_database(num\_of\_failures);  
dbms\_output.put\_line(num\_of\_failures);  
dbms\_dst.end\_upgrade(num\_of\_failures);  
dbms\_output.put\_line(num\_of\_failures);  
end;  
/

* Gather Workload Statistics

Gather system statistics during a regular workload period – otherwise inappropriate values for the CBO will be used:

SQL> EXECUTE dbms\_stats.gather\_system\_stats('start');

<< Run it for several hours on a workload >>

SQL> EXECUTE dbms\_stats.gather\_system\_stats('stop');

SQL> select pname NAME, pval1 VALUE, pval2 INFO

from aux\_stats$;

NAME VALUE INFO

-------------------- ---------- ------------------------------

STATUS COMPLETED

DSTART 04-03-2010 12:30

DSTOP 05-03-2010 12:30

FLAGS 1

CPUSPEEDNW 2498,65

IOSEEKTIM 11,405

IOTFRSPEED 25595,605

...

* Fixed Table Stats
  + Create fixed table statistics directly after catupgrd.sql has been completed:

SQL> exec DBMS\_STATS.GATHER\_FIXED\_OBJECT\_STATS;

* + Otherwise MMON may cause too much CPU load
  + Guidelines
    - Create fixed table statistics a week after upgrade
    - Gather fixed table statistics during a normal workload period
    - This task should be done only a few times per year

## Minimal downtime

### Transient Logical Standby

Concept:

* Build up a Physical Standby database
* Convert the Physical Standby into a Logical Standby
* Upgrade the Logical Standby database
* Switchover – Standby will be production system now
* Then: Flashback the former production database
* Convert it into a Physical Standby
* Upgrade just by log apply
* Eventually: Switchover to the original setup

