
ORACLE®

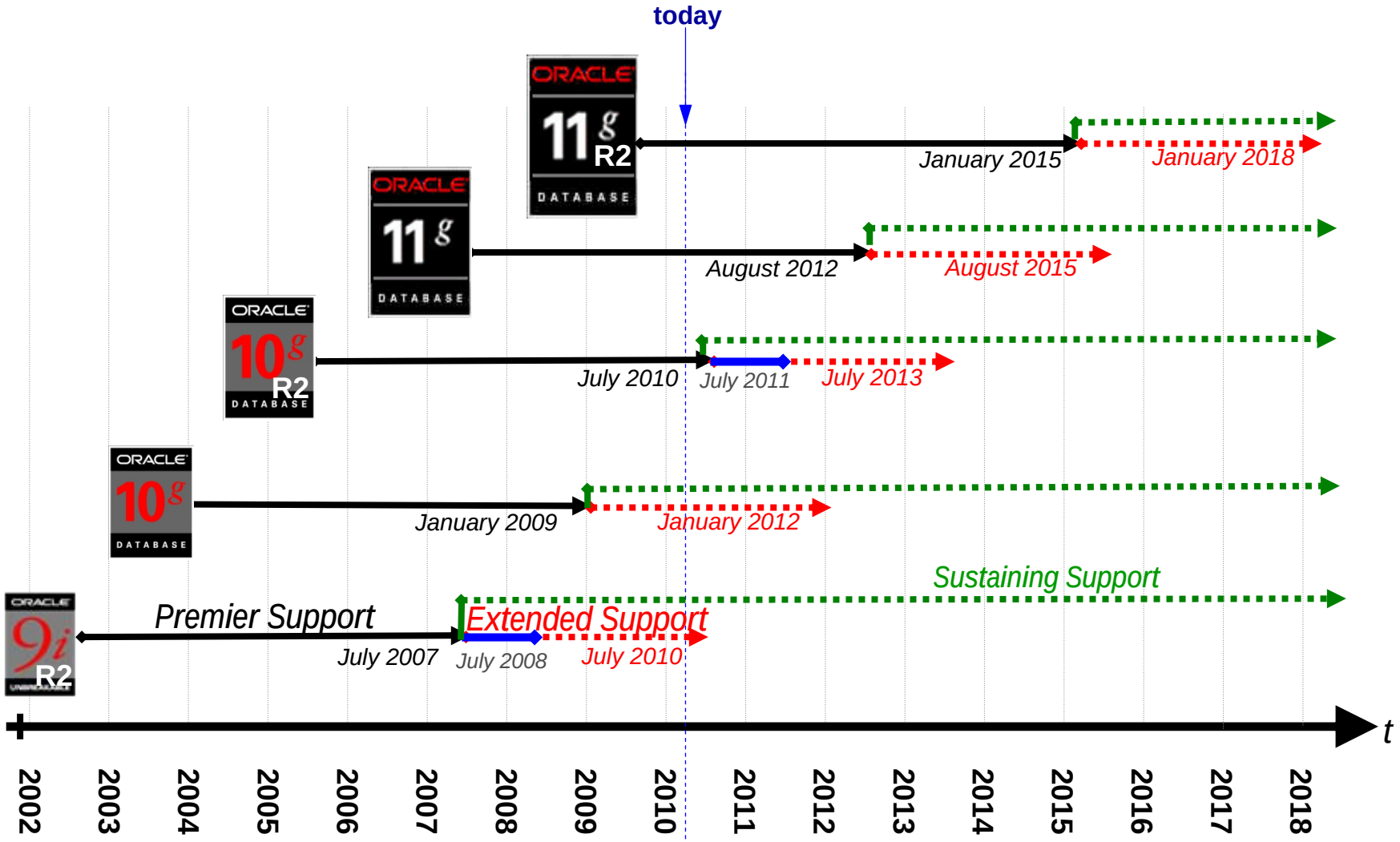
# Upgrading to 11g Database – Best Practices and Less Known Features

Morana Kobal Butković
Senior Sales Consultant
Oracle Hrvatska

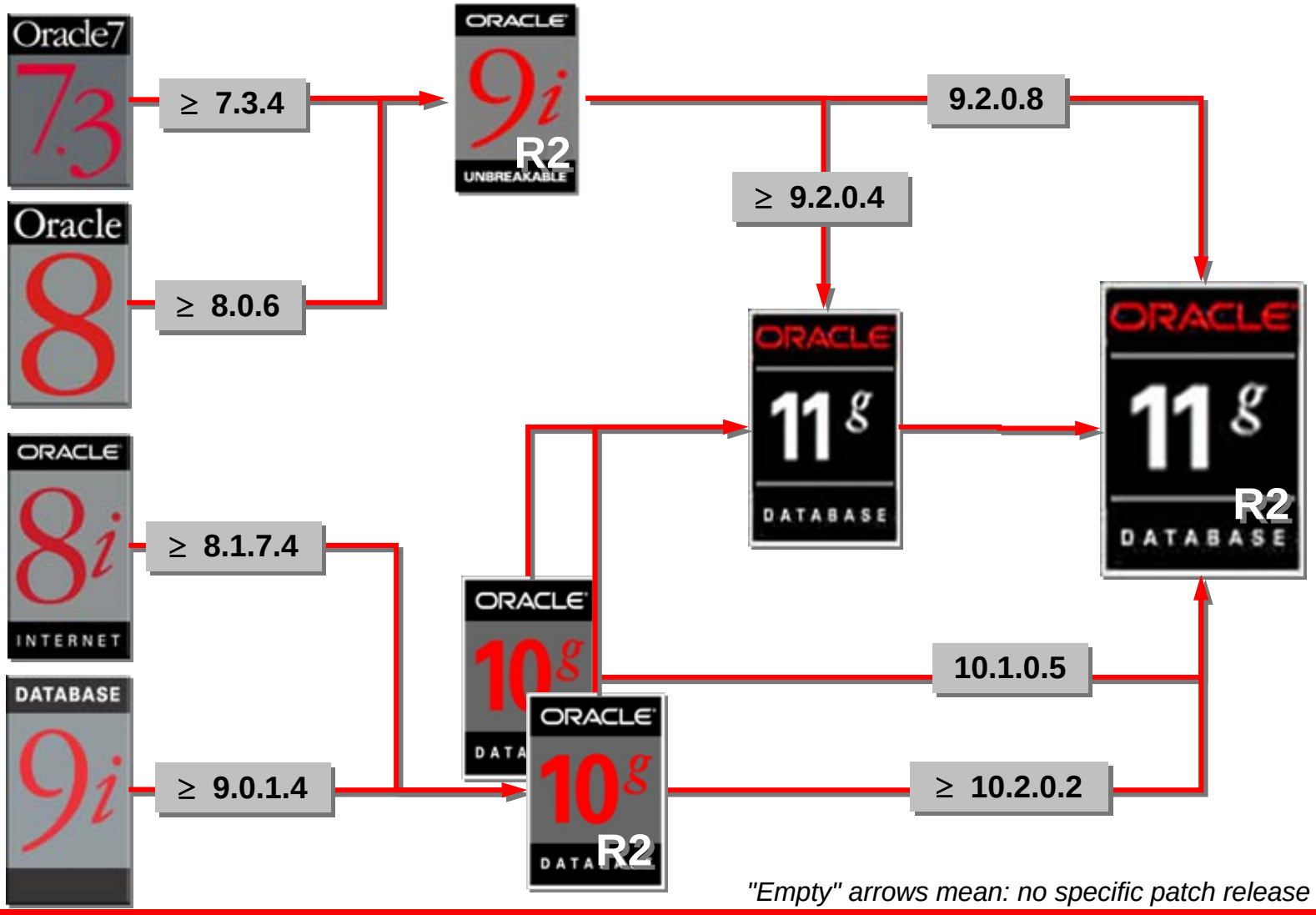
# Agenda

- Preparation
- Best Practices
- Performance Testing
- Demos

# Lifetime Support Policy

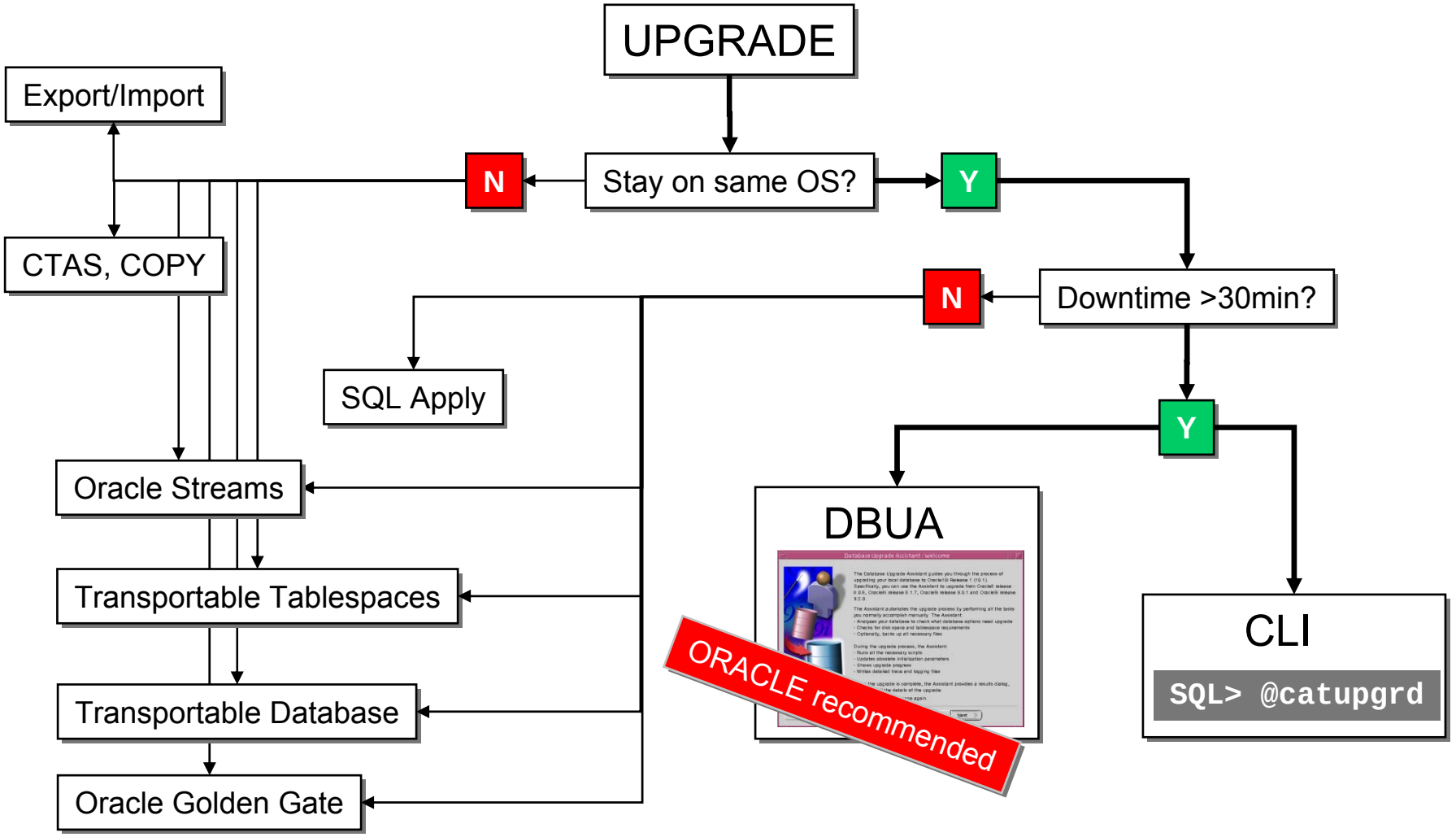


# Upgrade to Oracle Database 11*g*

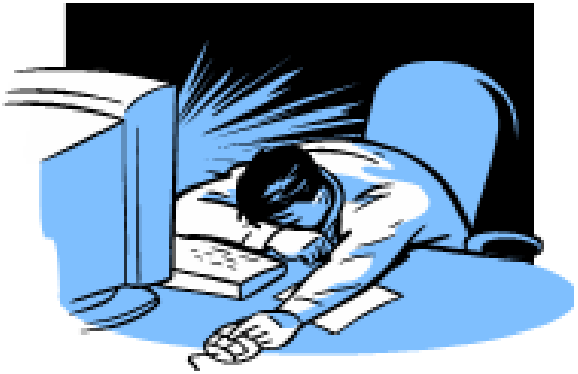



*"Empty" arrows mean: no specific patch release required*

# Different Ways To Upgrade



# Upgrade Length



- How long will it take to upgrade?
  - Usually between 30 and 90 minutes
  - Independent of:
    - Size of the database
    - Used datatypes
  - Dependent mainly on:
    - The number of installed components and options
    - Valid and non-stale data dictionary statistics
    - Number of synonyms – they'll get recompiled (upgrade from 9*i*)
    - Number of objects in XDB
    - To a lesser degree, if `COMPATIBLE` is increased:
      - Datafile headers are updated
      - Format of redo logs can change

# Upgrade Length

- Speed up your upgrade performance by:
  - Truncating the auditing table SYS.AUD$

```
SQL> truncate SYS.AUD$;
```

  - Creating dictionary statistics right before the upgrade
    - Oracle 9*i*:

```
SQL> exec DBMS_STATS.GATHER_SCHEMA_STATS
    ('SYS', options => 'GATHER',estimate_percent =>
     DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR
     ALL COLUMNS SIZE AUTO', cascade => TRUE);
```

    - Oracle 10*g*/11*g*:

```
SQL> exec DBMS_STATS.GATHER_DICTIONARY_STATS;
```

# 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 9.2.0.4 if upgrading to 11g and 9.2.0.8 if upgrading to 11g R2
- Migrating to a new hardware platform with same OS

- Consideration
  - Cannot upgrade to a system with a different operating system architecture
  - More manual steps required
  - Potential for errors due to typos, missed details
  - Upgrade scripts can be run again and again

# Command Line Upgrade

- Step-by-step:
  1. Complete online backup of the database
  2. Install 11*g* Oracle software and apply patch set, PSU etc.
  3. Analyze the DB using **`utlu112i.sql`** and follow all requirements given by the script
  4. Create a new 11*g* listener with NETCA
  5. Switch to the new environment, startup the DB (`startup upgrade`) and create the SYSAUX tablespace (only if source db is an Oracle 9*i* db)
  6. Run upgrade script **`catupgrd.sql`**
  7. Recompile with **`utlrp.sql`** - compare with **`utluiobj.sql`**
  8. Run **`catuppst.sql`** if you are upgrading from ≥10*g* for AWR
  9. Check the post upgrade status: **`utlu112s.sql`**

# Upgrade Mode

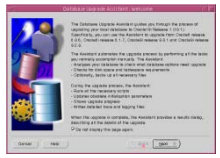
```
SQL> STARTUP UPGRADE;
```

```
ALTER SYSTEM SET _system_trig_enabled=FALSE SCOPE=MEMORY;
Autotune of undo retention is turned off.
ALTER SYSTEM SET _undo_autotune=FALSE SCOPE=MEMORY;
ALTER SYSTEM SET undo_retention=900 SCOPE=MEMORY;
ALTER SYSTEM SET aq_tm_processes=0 SCOPE=MEMORY;
ALTER SYSTEM SET enable_ddl_logging=FALSE SCOPE=MEMORY;
Resource Manager disabled during database migration: plan '' not set
ALTER SYSTEM SET resource_manager_plan='' SCOPE=MEMORY;
ALTER SYSTEM SET recyclebin='OFF' DEFERRED SCOPE=MEMORY;
Resource Manager disabled during database migration
```

*Taken from an example alert.log*

- Supresses unnecessary error messages like **`ORA-00942: table or view does not exist`** - thus logfiles will be easier to read and check

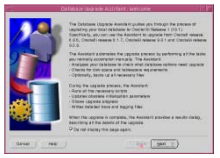
# When to Choose the DBUA



- 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
- Existing database is at least 9.2.0.4 if upgrading to 11g or 9.2.0.8 for 11g R2
- Note: especially useful for RAC and ASM installations*

- Consideration:
  - Source and target Oracle Homes must be on the same system
  - Cannot be re-run if an error is encountered mid-upgrade

# Database Upgrade Assistant (GUI)



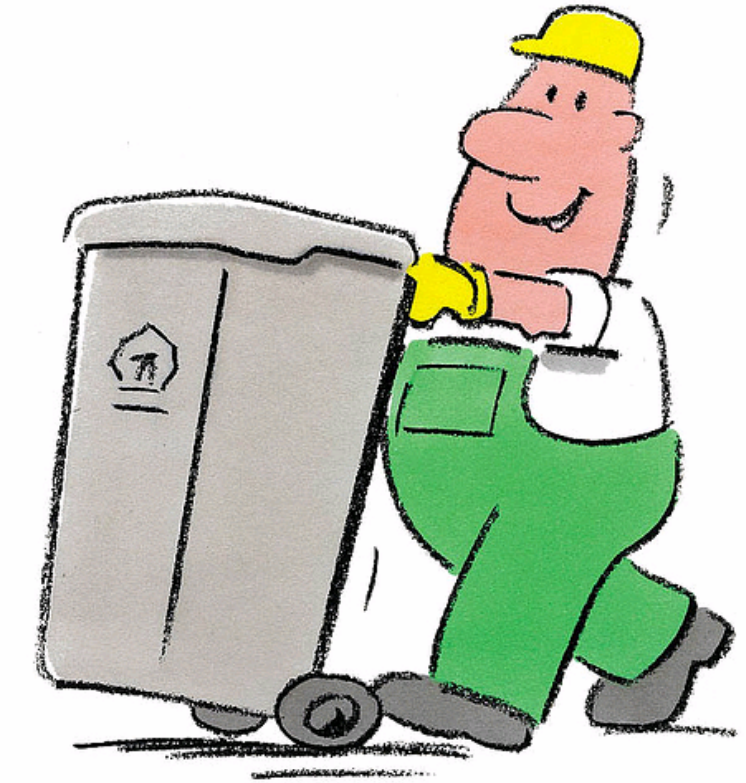
- Features:
  - Graphically led upgrade
  - Lots of important checks
  - RAC *aware* - inclusion of all nodes
    - for RAC (almost) a must !!!
  - Offline Backup and Restore possible
  - ASM upgrade (until 11.1)
  - Oracle XE upgrade to SE & EE
  - Patch upgrades
  - Best Practice: Before you start DBUA
    - Run `$OH_11g/rdbms/admin/utlu112i.sql` in your current environment
  - Logs:
    - `$ORACLE_HOME/cfgtoollogs/dbua`
  - Documented in Chapter 3 of the Oracle® Database Upgrade Guide

# Agenda

- Preparation
- Best Practices
- Performance Testing
- Upgrade Summary

# Best Practice

- Sanity operations:  important checks

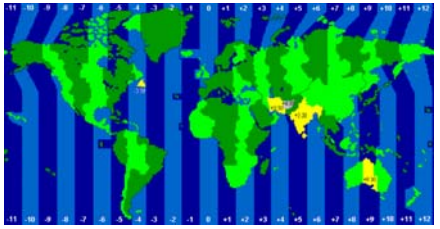


# Recycle bin

- If upgrading from 10*g* or 11*g*, purge the recycle bin before the upgrade.

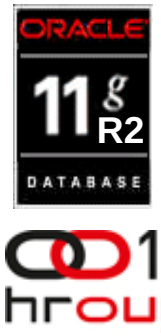
```
SQL> purge DBA_RECYCLEBIN;
```

# Timezone Patches



- Why DST timezone patches? (DST: Daylight Savings Time)
  - Since 2007 DST definitions and timezone names have been changed several times
  - Timezone versions by release:
    - Oracle 9*i*: TZ V1
    - Oracle 10.1: TZ V2
    - Oracle 10.2.0.1/2: TZ V2
    - Oracle 10.2.0.3: TZ V3
    - Oracle 10.2.0.4: TZ V4
    - Oracle 11.1: TZ V4
      - Source release needs to be patched to TZ V4 – otherwise no upgrade will be possible
    - Oracle 11.2: TZ V11
      - Source release does not have to be patched. Timezone conversion will be done in 11.2

# Timezone Patches - 11*g Release 2*




- Upgrade to Oracle Database 11*g* Release 2:
  - New 11.2-$OH has timezone V11
  - No need to patch the source $OH
  - Database only needs to be adjusted if you are using the datatype `TIMESTAMP WITH TIMEZONE`
    - Conversion done after the upgrade
    - See Note 944122.1
    - Package **`DBMS_DST`**
      - `DBMS_DST.FIND_AFFECTED_TABLES`
      - `DBMS_DST.BEGIN_UPGRADE`
      - `DBMS_DST.UPGRADE_DATABASE`
      - `DBMS_DST.END_UPGRADE`

# Best Practice

- Always run the pre-upgrade script:
  - Upgrade to Oracle Database 11g : **`utlu111i.sql`**
  - Upgrade to Oracle Database 11.2 : **`utlu112i.sql`**

# Pre-Upgrade Check

- Run `utlu112i.sql` in your current environment

```
Oracle Database 11.2 Pre-Upgrade Information Tool    09-21-2009 22:33:20

**********************************************************************
Database:
**********************************************************************
--> name:          ORCL
--> version:       10.2.0.3.0
--> compatible:    10.2.0.3.0
--> blocksize:     8192
--> platform:      Linux IA (32-bit)
--> timezone file: V4

[..]

**********************************************************************
Update Parameters: [Update Oracle Database 11.2 init.ora or spfile]
**********************************************************************
WARNING: --> "java_pool_size" needs to be increased to at least 64 MB

[..]

**********************************************************************
Miscellaneous Warnings
**********************************************************************
WARNING: --> Database is using a timezone file older than version 11.
.... After the release migration, it is recommended that DBMS_DST package
.... be used to upgrade the 10.2.0.3.0 database timezone version
.... to the latest version which comes with the new release.
```

# Pre-Upgrade Check

- Get the current version of `utlu1`*`nm`*`i.sql`
  - Download it
  - Note:884522.1

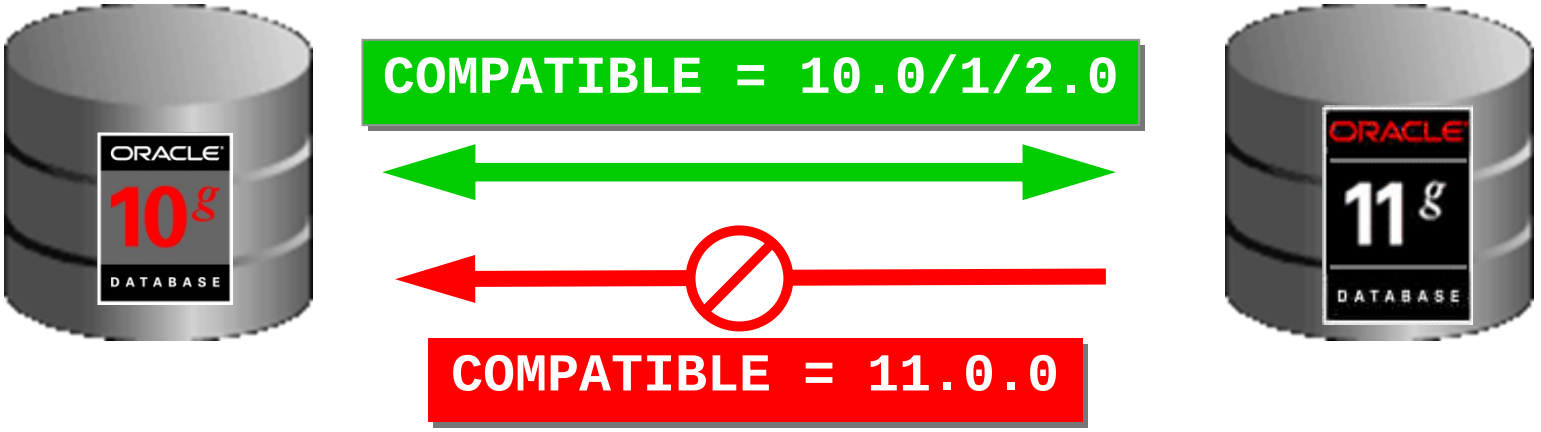
| Coming From Version | Upgrade Target Version |
|---|---|
| 9.2.0 (9.2.0.8 and beyond),<br>10.1.0, 10.2.0,<br>11.1.0 | 11gR2 - utlu112i.sql |
| 9.2.0 (9.2.0.4 and beyond),<br>10.1.0,10.2.0 | 11gR1- utlu111i.sql |
| 8.1.7, 9.0.1,<br>9.2.0 (9.2.0.4 and beyond),<br>10.1.0 | 10gR2 - utlu102i.sql |

# Best Practice

- After upgrade leave COMPATIBLE at the original value for a week before changing to 11.1 or 11.2.

# Parameter COMPATIBLE

- COMPATIBLE has to be at least 10.1.0 for an 11*g* database
- No way back once ≥11.1.0 has been enabled
  - Supported release downgrade to 10.1.0.5, ≥10.2.0.2, ≥11.1.0.6
  - No **`ALTER DATABASE RESET COMPATIBILITY`** command anymore



# Parameter COMPATIBLE

- DBUA raises `COMPATIBLE` only for 9*i* databases
- To enable new features after the upgrade:
  - 11.1:

```
SQL> alter system
set compatible='11.1.0' scope=spfile;
```

  - 11.*2* :

```
SQL> alter system
set compatible='11.2.0' scope=spfile;
```

  - Afterwards: restart the database
    - New features will be enabled
    - Datafile headers will be adjusted
    - Redologfiles formats will be adjusted during first access

# Best Practice

- Do you have a fallback strategy? ...

# Fallback Strategies

- Always:
  - Create a valid and complete online backup with RMAN
  - Test the restore and the recovery at least once!!!
- Downgrade Options:
  - Back to Oracle Database 10*g*/11*g*
    - Use the downgrade scripts `catdwgrd.sql` and `catrelod.sql`
      - See the *Database Upgrade Guide, Chapter 6* and Note:443890.1
    - Datapump with VERSION parameter (COMPATIBLE can be raised)
  - Back to Oracle Database 9*i*
    - Export/import
    - Use 9*i* `exp` to extract the data and 9*i* `imp` to import the data back
    - Note:158845.1

# Fallback Strategy: catdwgrd.sql

- Downgrade with `catdwgrd.sql`
  - Note:443890.1
    - Downgrade to the release you've upgraded from
      - 10.1.0.5
      - 10.2.0.2/3/4
      - 11.1.0.6/7
    - Only possible **if COMPATIBLE hasn't been raised**!!!
      - Please note:
        A downgrade will only be possible to the release you've upgraded from - so if a patch set has been applied always apply it before the upgrade starts - otherwise you'll only be able to downgrade to the release you've patched

# Fallback Strategy: catdwgrd.sql

- Downgrade with `catdwgrd.sql` to 10g
  - Task in 11g environment:

```
SQL> SPOOL /tmp/downgrade.log
SQL> STARTUP DOWNGRADE
SQL> @catdwgrd.sql
SQL> SPOOL OFF
```

  - Switch to your pre-upgrade 10*g* environment:

```
SQL> STARTUP UPGRADE
SQL> SPOOL /tmp/reload.log
SQL> @catrelod.sql
-- The catrelod.sql script reloads the appropriate version of
-- all of the database components in the downgraded database.
SQL> SPOOL OFF
```

    - Please note: additional steps are required if EM repository resides in the database - please see chapter 6 ***Downgrading a Database*** in the Oracle 11g Upgrade Guide

# Best Practice

- After the upgrade ...

# Post Upgrade

- Create system statistics during a regular workload period - otherwise non-appropriate values for the CBO will be used:

```
SQL> exec DBMS_STATS.GATHER_SYSTEM_STATS('start');
...  -- some time delay while the database is under a typical workload execute
SQL> exec DBMS_STATS.GATHER_SYSTEM_STATS('stop');
```

```
SQL> select pname NAME, pval1 VALUE, pval2 INFO
     from aux_stats$;

NAME                      VALUE INFO
------------------ ---------- ------------------------------
STATUS                          COMPLETED
DSTART                          04-03-2009 12:30
DSTOP                           05-03-2009 12:30
FLAGS                        1
CPUSPEEDNW            1392.39
IOSEEKTIM                8.405
IOTFRSPEED           255945.605
...
```

# Post Upgrade

- Create fixed table (X$) statistics
  - Directly after `catupgrd.sql` has been completed
    - This will speed up the job processing for recompilation with

```
SQL> exec DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

  - Again: after a few days regular database workload

# Agenda

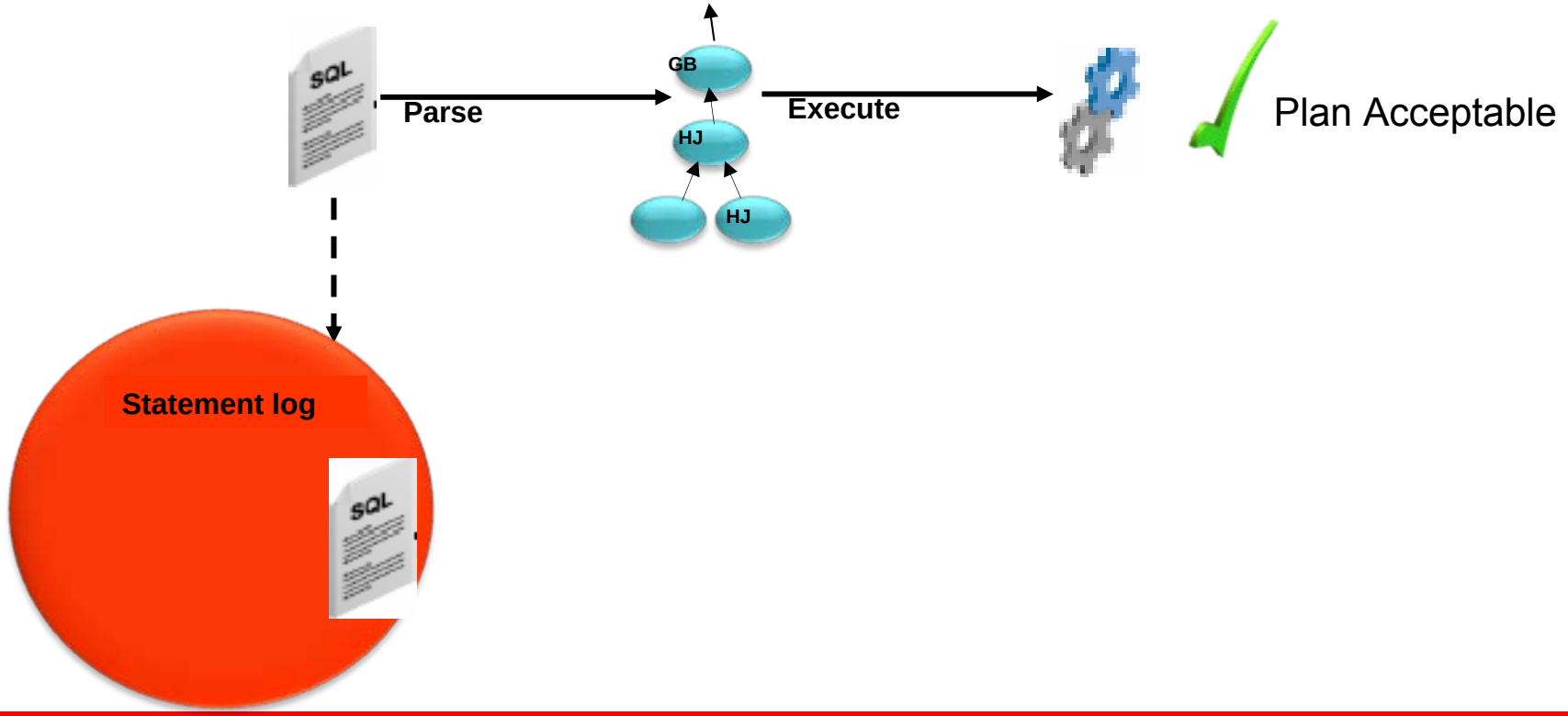
- Preparation
- Best Practices
- Performance Testing
- Demos

# Prevent execution plan changes

- Classical approach:
  - Rule Based Optimizer (RBO desupport since Oracle 10*g* - Note:189702.1)
  - Hints
  - Stored Outlines
  - Rewriting SQL statements
  - `optimizer_features_enabled=n.n.n`
  - Change specific optimizer parameters
  - Import and lock object and systems statistics
- Modern, efficient and better resource consumption:
  - SQL Plan Management

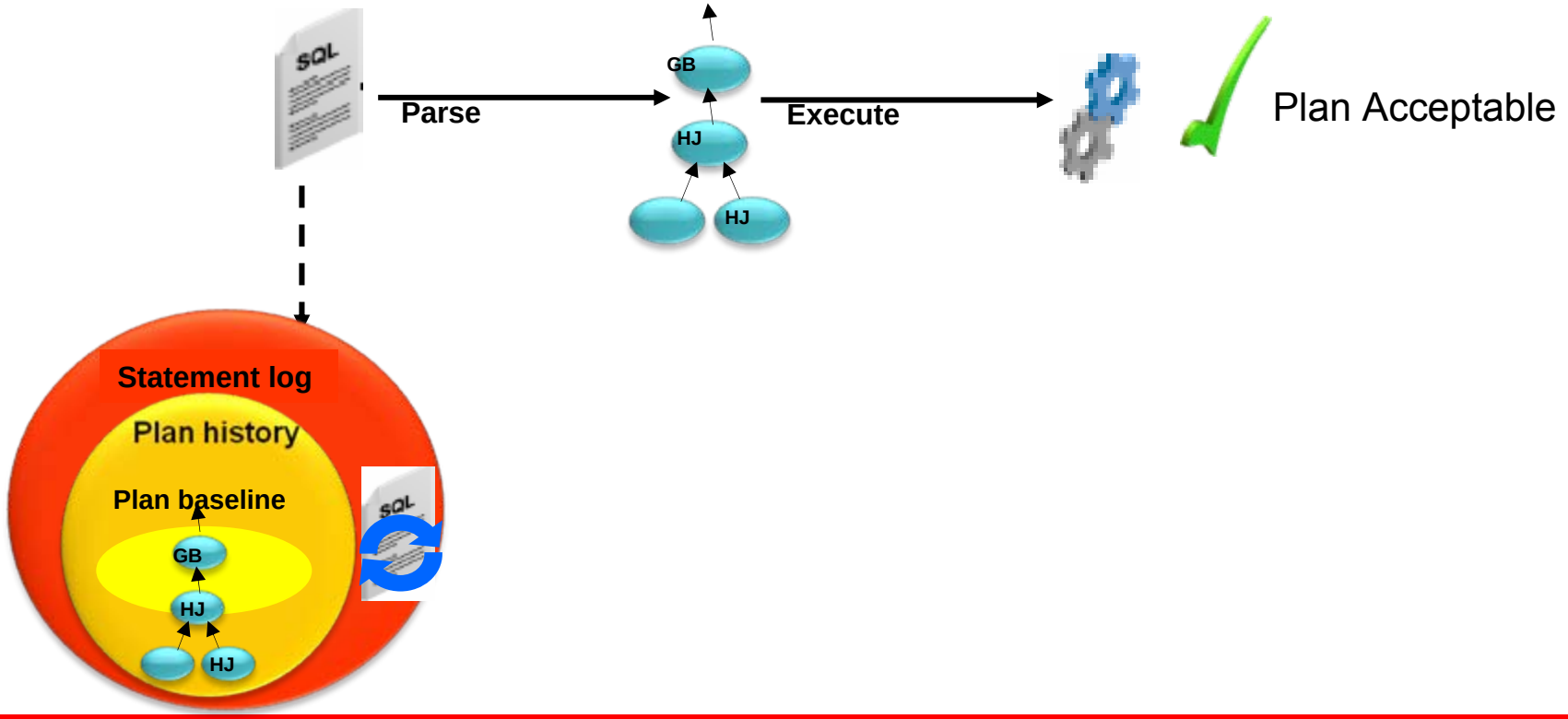
# With SQL Plan Management

- SQL statement is parsed for the first time and a plan is generated
- Check the log to see if this is a repeatable SQL statement
- Add SQL statement signature to the log and execute it
- Plan performance is still “verified by execution”



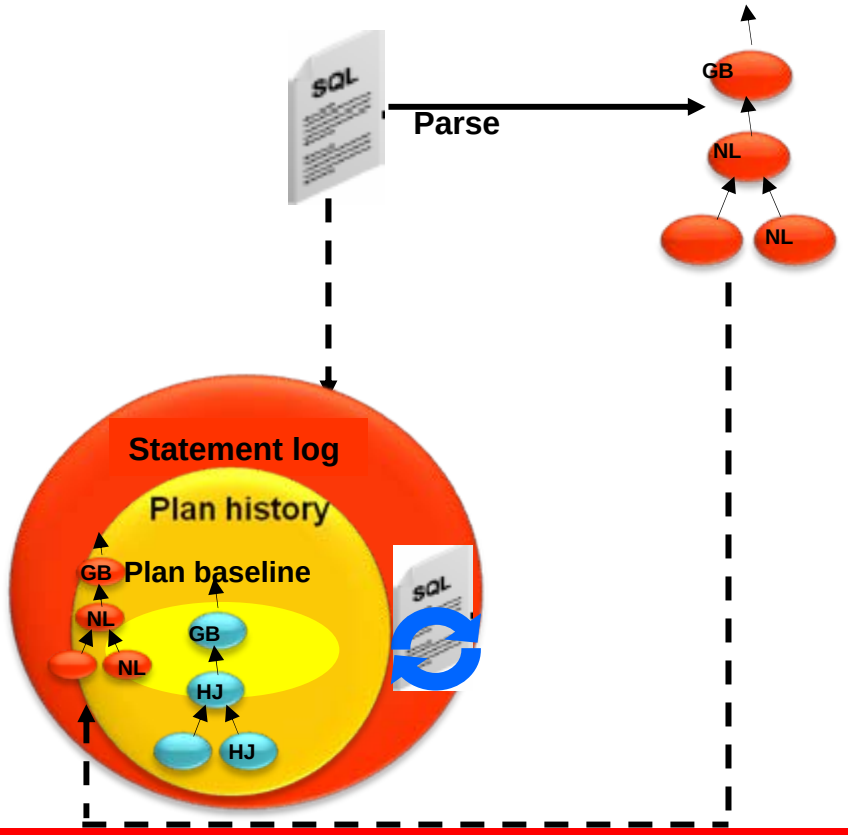
# With SQL Plan Management

- SQL statement is parsed again and a plan is generated
- Check log to see if this is a repeatable SQL statement
- Create a Plan history and use current plan as SQL plan baseline
- Plan performance is “verified by execution”



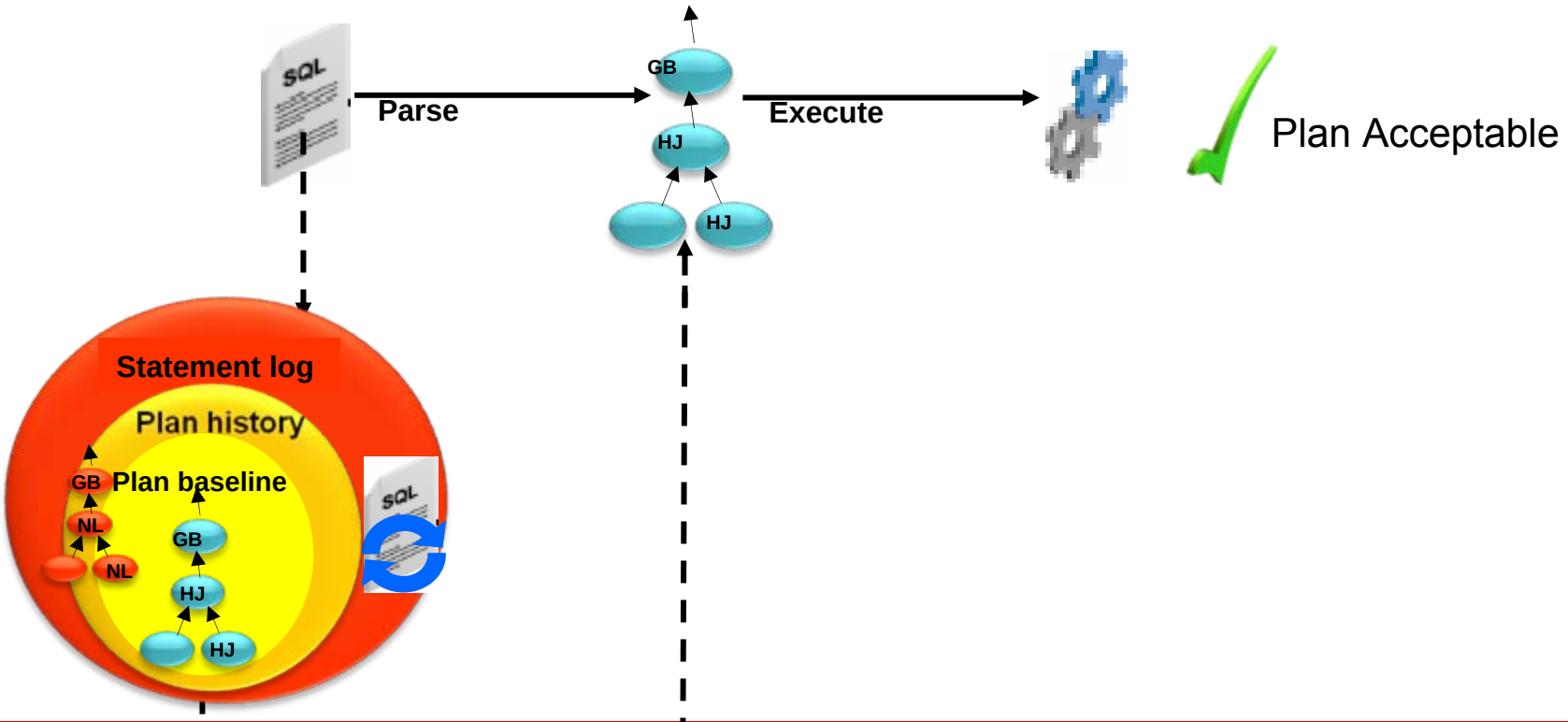
# With SQL Plan Management

- Something changes in the environment
- SQL statement is parsed again and a **new plan is generated**
- New plan is not the same as the baseline – **new plan is not executed** but marked for verification



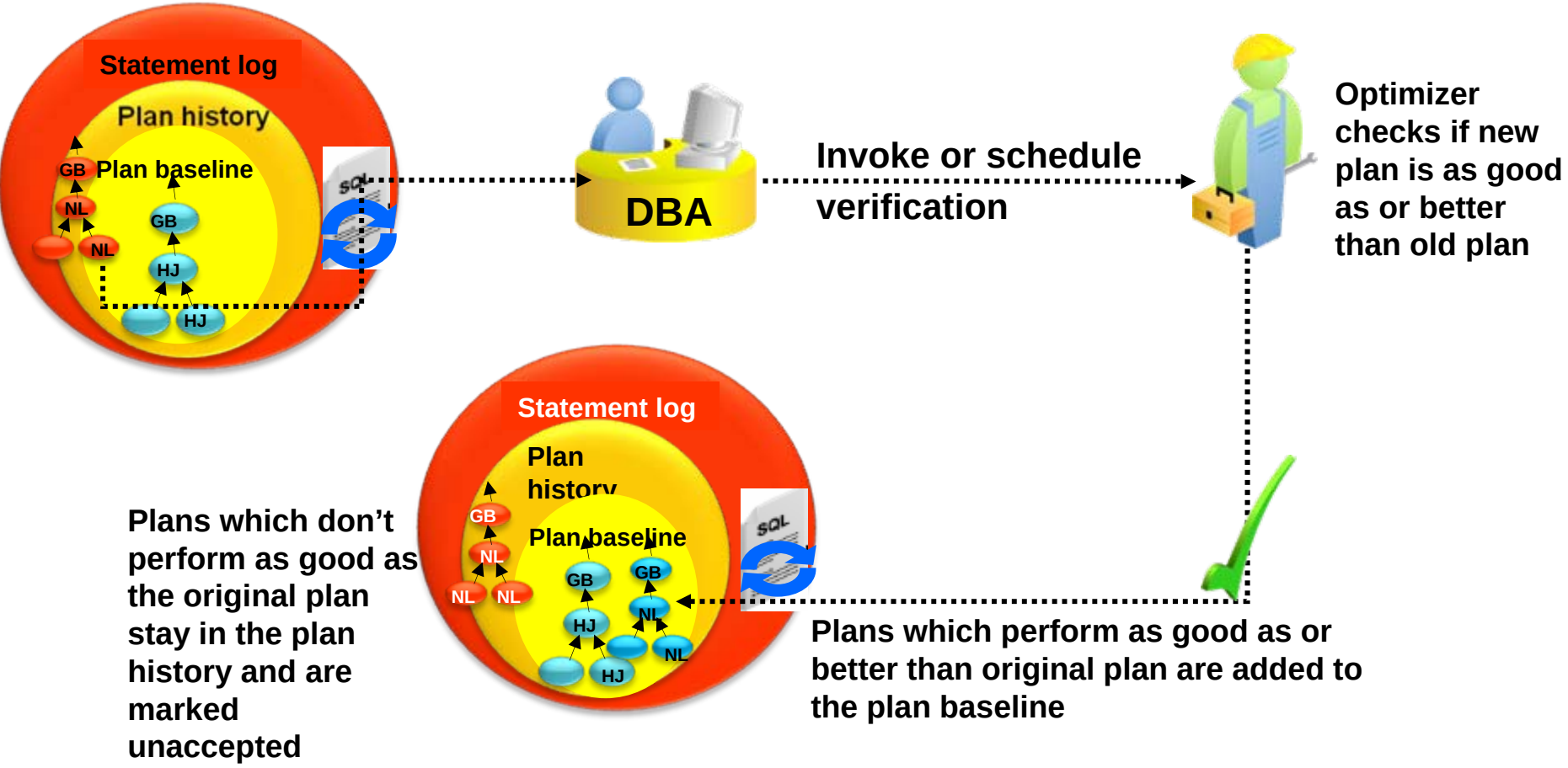
# With SQL Plan Management

- Something changes in the environment
- SQL statement is parsed again and a **new plan is generated**
- New plan is not the same as the baseline – **new plan is not executed** but marked for verification
- Execute known plan baseline - plan performance is “verify by history”



# Verifying the new plan

- Non-baseline plans will not be used until verified
- DBA can verify plan at any time



# SQL Plan Management – the details

- Controlled by two init.ora parameter
  - ***optimizer_capture_sql_plan_baselines***
    - Controls auto-capture of SQL plan baselines for repeatable stmts
    - Set to FALSE by default in 11gR1
  - ***optimizer_use_sql_plan_baselines***
    - Controls the use of existing SQL plan baselines by the optimizer
    - Set to TRUE by default in 11gR1
- Monitoring SPM
  - Dictionary view DBA_SQL_PLAN_BASELINE
  - Via the SQL Plan Control in EM DBControl
- Managing SPM
  - PL/SQL package DBMS_SPM or via SQL Plan Control in EM DBControl
  - Requires the 'administer sql management object' privilege

# SPM Plan Capture – Bulk

- From SQL Tuning Set (STS)
  - Captures plan details for a (critical) set of SQL Statement in STS
  - Load these plans into SPM as baseline plans
- From Stored Outlines
  - Migrate previously created Stored Outlines to SQL plan baselines
- From Cursor Cache
  - Load plans from the cursor cache into SPM as baseline plans
    - Filters can be specified (SQL_ID, Module name, schema)
- From staging table
  - SQL plan baselines can be captured on another system
  - Exported via a table (similar to statistics) and imported locally
  - Plan are “unpacked” from the table and loaded into SPM

# Real Application Testing

- Goal:
  - Record and replay a real workload to see how the new system performs
  - Find regressions and changing plans **before** the upgrade

- Licensable database pack "Real Application Testing"
  - ⇨ Available since Oracle Database 11.1.0.6
  - ⇨ Available with patch set 10.2.0.4
  - ⇨ Available as single patch for 9.2.0.8 and 10.2.0.2/3
    - ⇨ For patch numbers please see Note:560977.1

# Oracle Real Application Testing

## Database Replay

- Replay actual production database workload in test environment
- Identify, analyze and fix potential instabilities before making changes to production
- Capture Workload in Production
  - Capture full production workload with real load, timing & concurrency characteristics
  - Move the captured workload to test system
- Replay Workload in Test
  - Make the desired changes in test system
  - Replay workload with full production characteristics
  - Honor commit ordering
- Analyze & Report
  - Errors
  - Data divergence
  - Performance divergence

## SQL Performance Analyzer

- Enables identification of SQL performance regressions ***before*** end-users can be impacted
- SPA can help with any change that impacts SQL execution plan
  - DB upgrades
  - Optimizer statistics refresh
  - New indexes, Materialized Views, Partitions, etc.
- Automates SQL performance tracking of hundreds of thousands of SQL statements – impossible to do manually
- Captures SQL workload with low overhead
- Integrated with SQL Tuning Advisor and SQL Plan Baselines for regression remediation

# Real Application Testing

- Real Application Testing consists of:
  - Database Replay
    - Package **DBMS_WORKLOAD_CAPTURE**
      - ⇨ Capture works in 9.2.0.8 and 10.2.0.2/3/4 and 11.1.0.x and 11.2.0.x
    - Package **DBMS_WORKLOAD_REPLAY**
      - ⇨ Replay works in 11.1.0.x and 11.2.0.x
  - SQL Performance Analyzer (SPA)
    - Package **DBMS_SQLPA**
      - ⇨ Collecting statements works in:
        - ⇨ 9.2.0.x and 10.1. 0.x with sql tracing
        - ⇨ 10.2.0.2/3/4 and 11.1.0.x and 11.2.0.x by capturing from cursor cache
      - ⇨ Evaluation and comparison works with:
        - ⇨ 10.2.0.2/3/4 and 11.1.0.x and 11.2.0.x
  - SQL Tuning Sets (STS)
    - Package **DBMS_SQLTUNE**

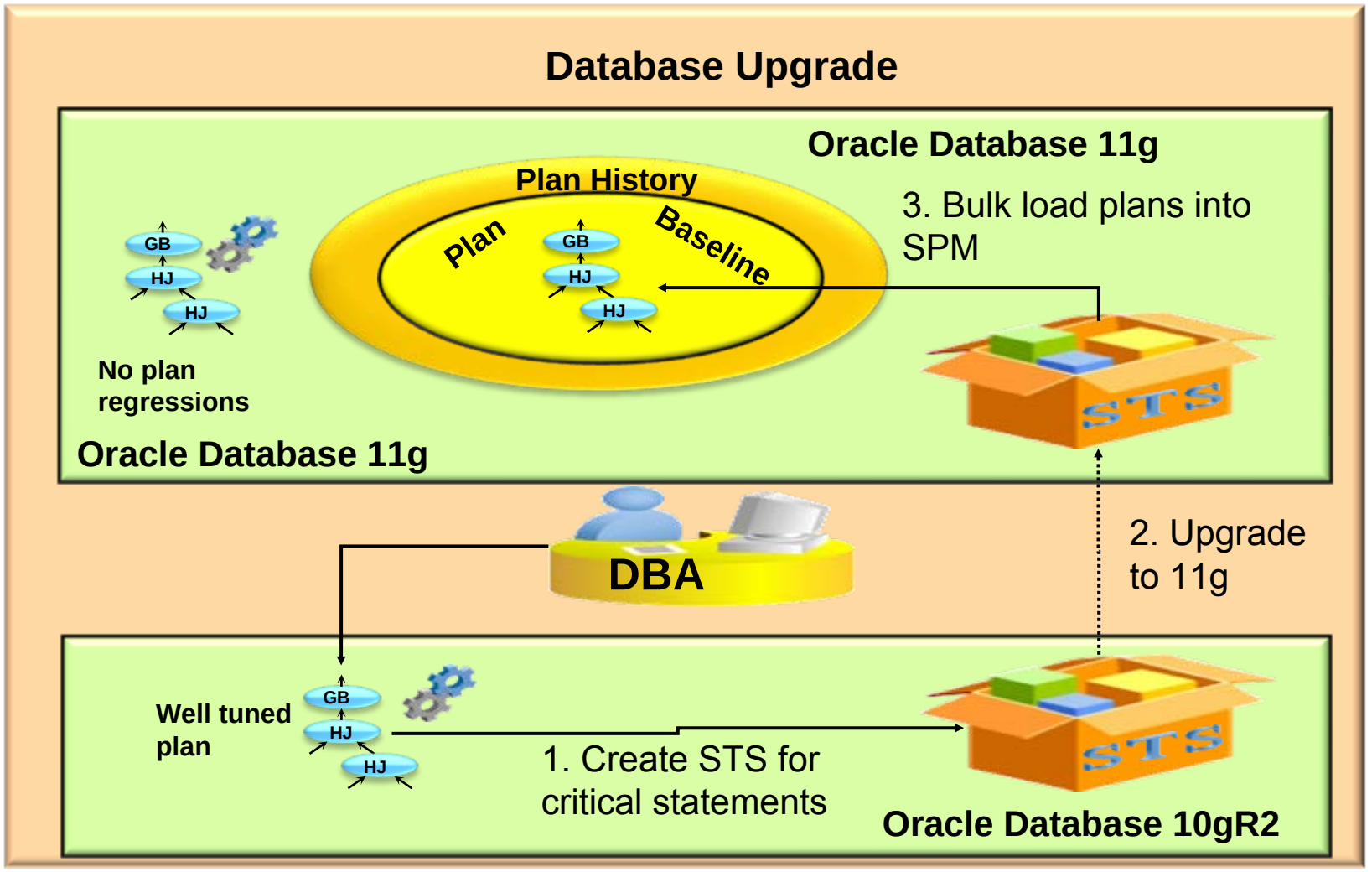
# Testing Pre-Upgrade Steps

- Testing on the new Database Release
  - Use hardware identical to product
  - Use a copy of the ‘live’ data from product
  - Ensure all important queries and reports are tested
  - Capture all necessary performance information during tests
  - Ensure comparable test results are available for your current Oracle release
- Capture current 10g execution plans
  - Using SQL Performance Analyzer
  - Using Stored Outlines
  - Using SQL Tuning Sets
  - Using exported SQL plan baselines

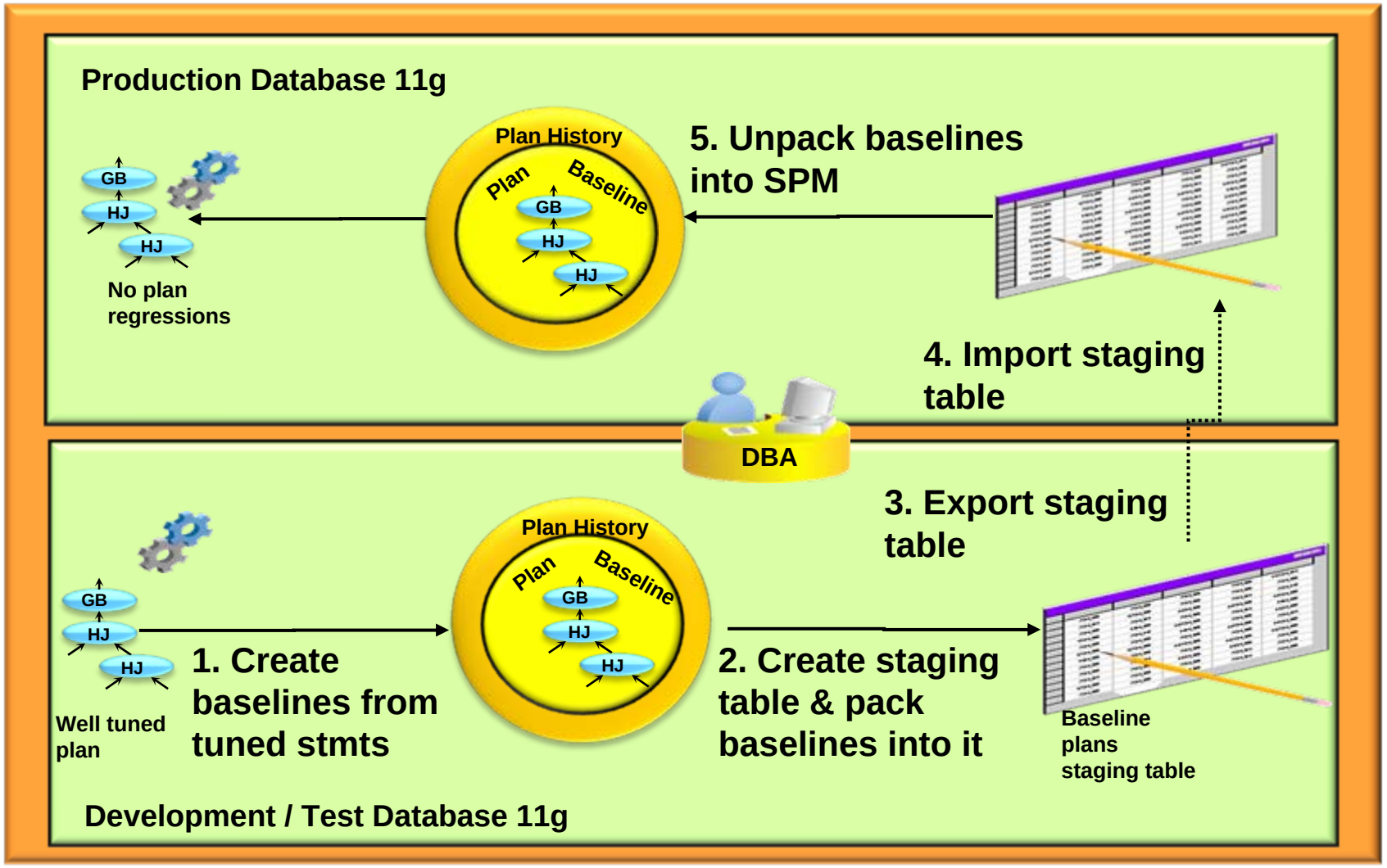
# Testing on the new database release
## *Removing old Optimizer hints*

- If there are hints for every aspect of the execution plan the plan won't change between releases (Stored Outline)
- Partial hints that worked in one release may not work in another
- Test all SQL stmts with hints on the new release using the parameter _optimizer_ignore_hints=TRUE
  - Chance are the SQL stmts will perform better without any hints

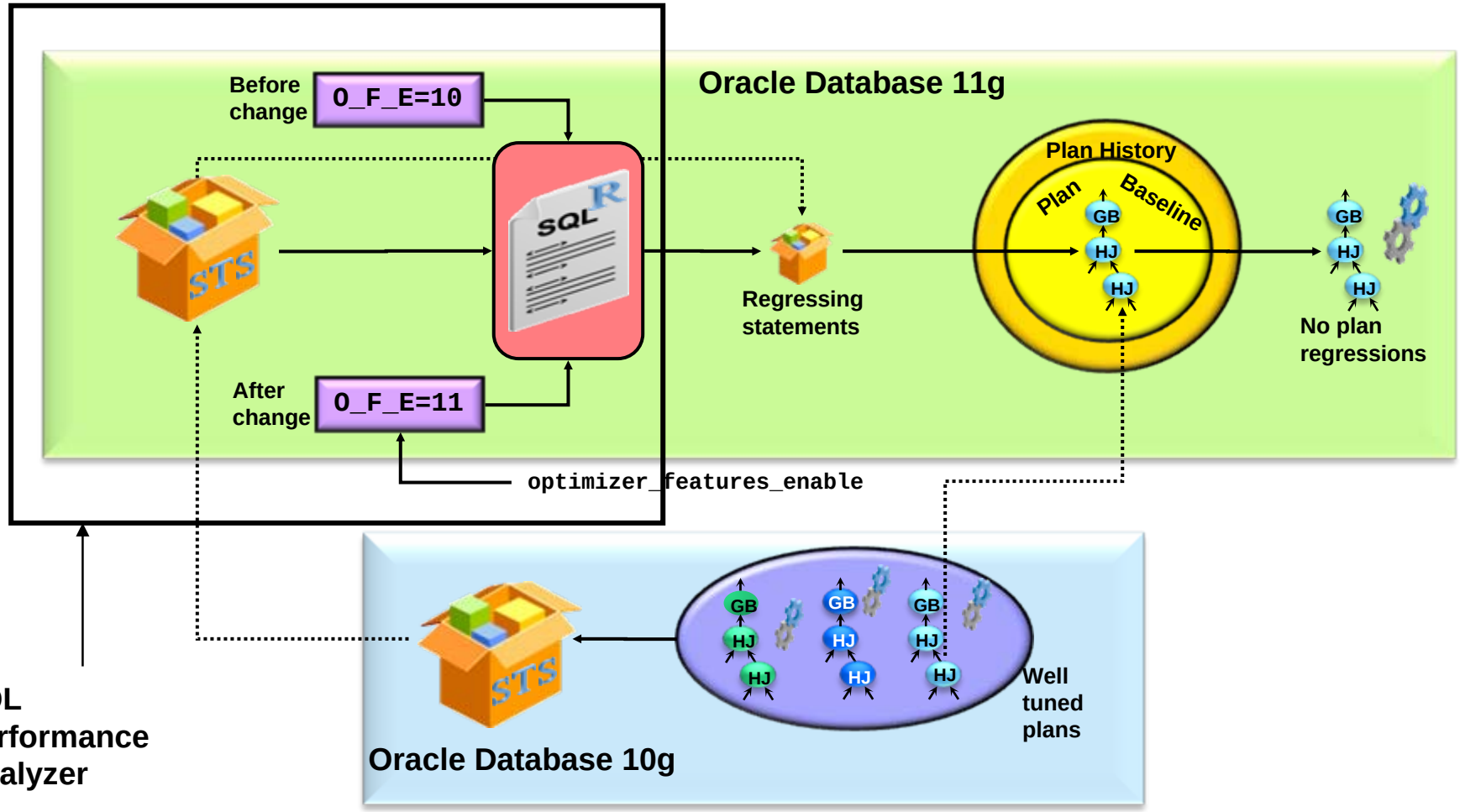
# Capturing Plans using SQL Tuning Set



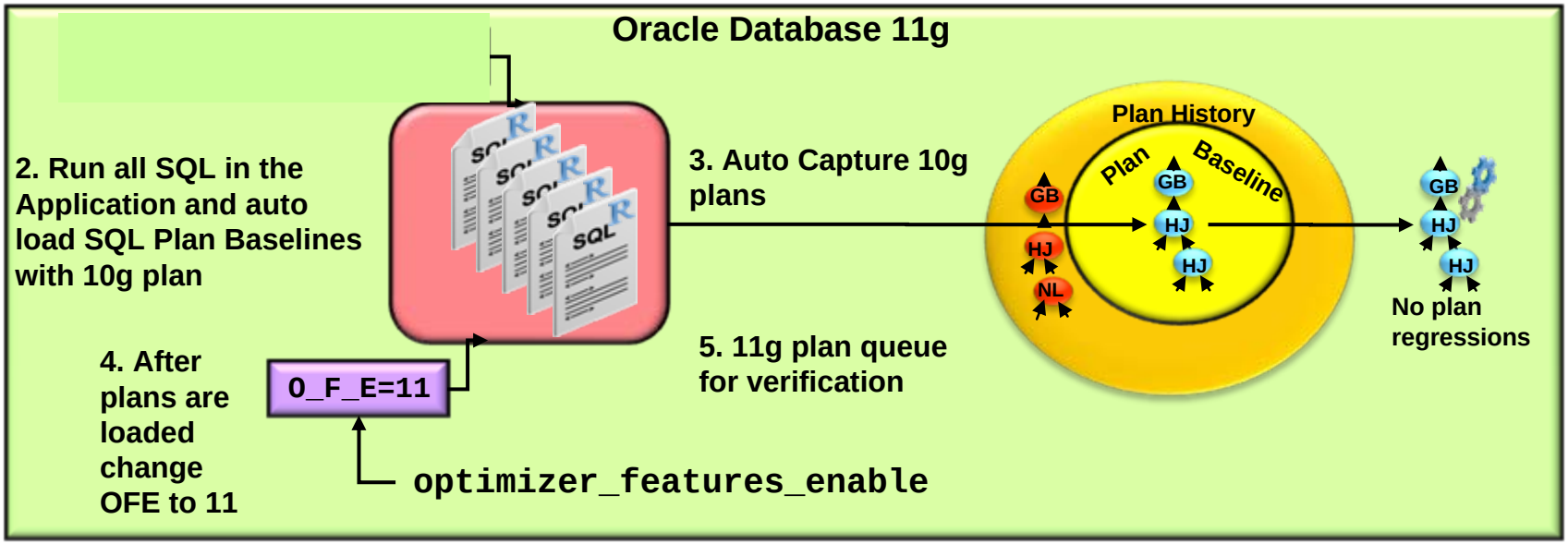
# Capturing Plans using an 11g test env



# Capturing Plans using SPA



# SQL Plan Management - general upgrade strategy




- Seeding the SQL Plan Baselines with 10g plans No plan change on upgrade
- After all SQL Plan Baselines are populated switch Optimizer_Features_Enable to 11g
  - new 11g plans will only be used after they have been verified

# What to do with statistics after upgrade

- Use last known 10g stats until system is stable
- Switch on incremental statistics for partitioned tables
  - `DBMS_STATS.SET_GLOBAL_PREFS('INCREMENTAL','TRUE');`
- Temporarily switch on pending statistics
  - `DBMS_STATS.SET_GLOBAL_PREFS('PENDING','TRUE');`
- Gather 11g statistics
  - `DBMS_STATS.GATHER_TABLE_STATS('sh','SALES');`
- Test your critical SQL statement with the pending stats
  - `Alter session set optimizer_use_pending_statistics=TRUE;`
- When proven publish the 11g statistics
  - `DBMS_STATS.PUBLISH_PENDING_STATS();`

# Agenda

- Preparation
- Best Practices
- External References
- Demos

# Demo SQL Performance Analyzer & DB Replay for Upgrade 10.2 ->11g


ORACLE®