

# ORACLE®

## **Take the Guesswork out of SQL Performance with SPA**

Mughees A. Minhas  
Director of Product Management  
Database and Systems Management

# Outline

- Motivation
- Overview
- Use Cases
- Real-world Deployments
- Conclusion

# Motivation

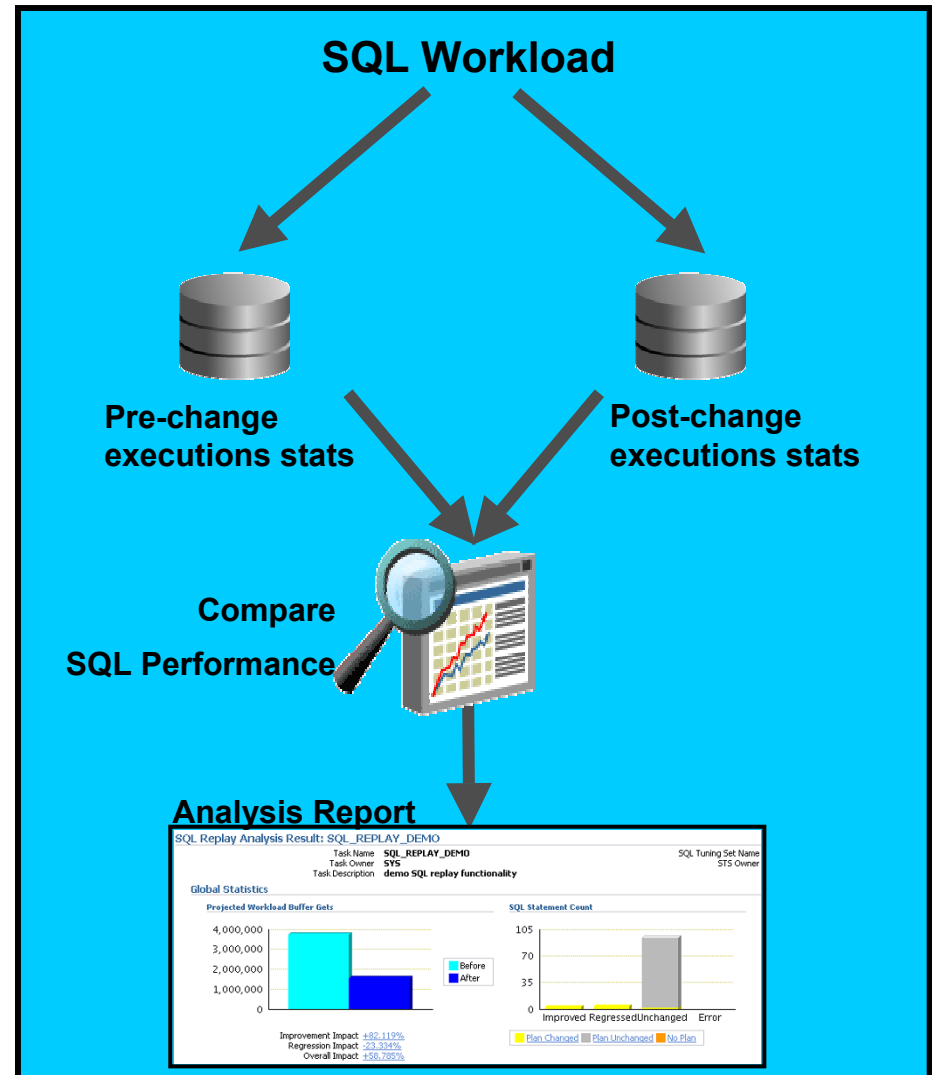
- Businesses want systems that are performant and meet SLA's
- SQL performance regressions are #1 cause of poor system performance
- Solution for proactively detecting all SQL regressions resulting from changes not available
- DBA's use ineffective and time-consuming manual scripts to identify problems



**SPA identifies all changes in SQL performance before end-users can be impacted**

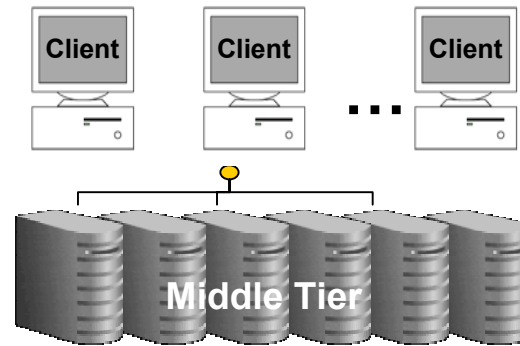
# SQL Performance Analyzer (SPA)

- Test impact of change on SQL query performance
- Capture SQL workload in production including statistics & bind variables
- Re-execute SQL queries in test environment
- Automatically remediate regressed SQL
  - Integrated with SQL Plan Baselines and SQL Tuning Advisor to provide end-to-end solution



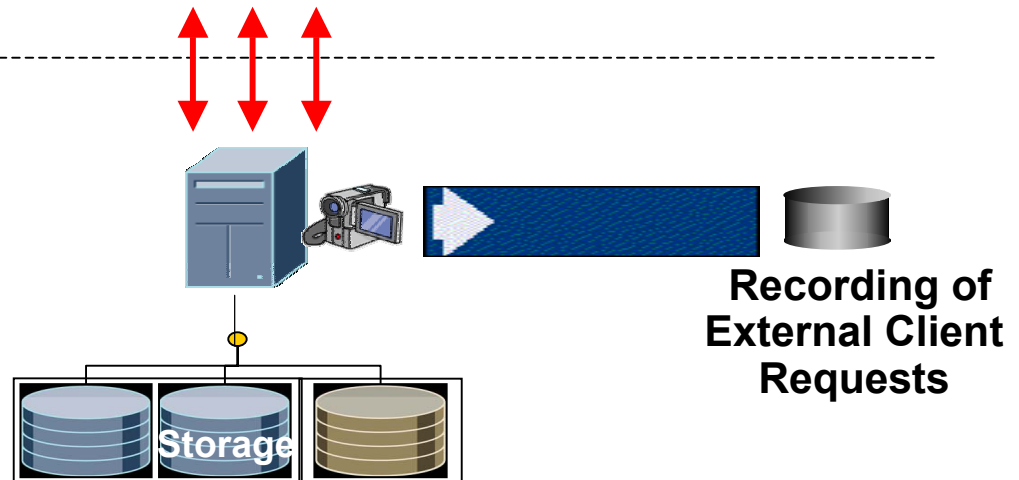
# SPA: Supported Changes

Changes  
Unsupported



## Changes Supported

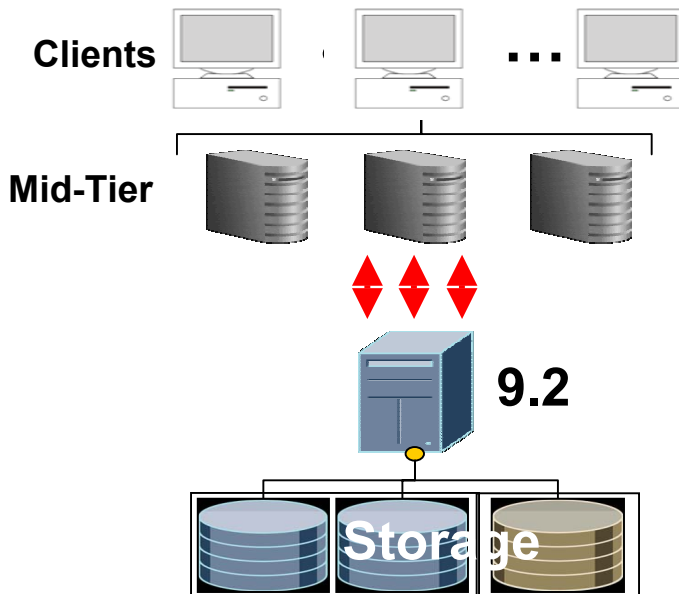
- Database Upgrades, Patches
  - Schema, Parameters
- Optimizer statistics refresh
  - Tuning actions
- I/O subsystem changes
  - Etc.



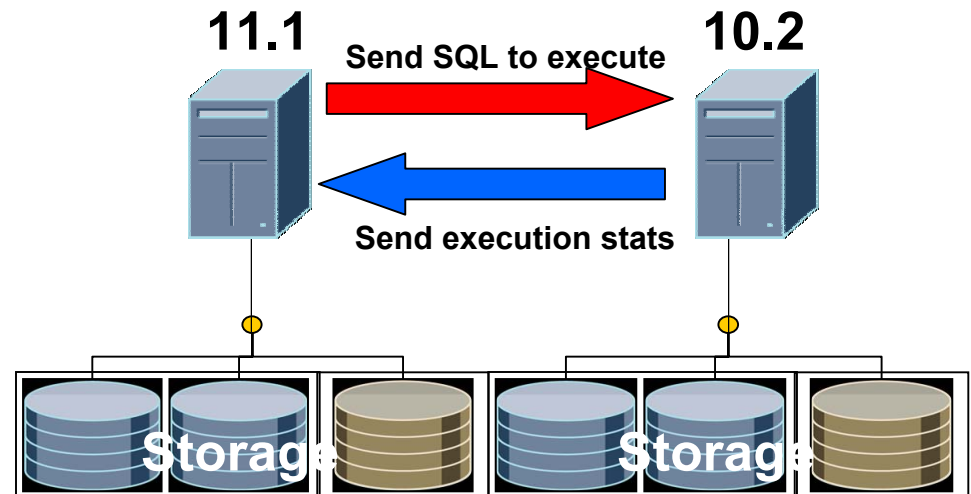
# SQL Performance Analyzer

Upgrading from Oracle Database 9.2 to 10.2

Capture from 9.2 or higher



Test in 10.2 or higher



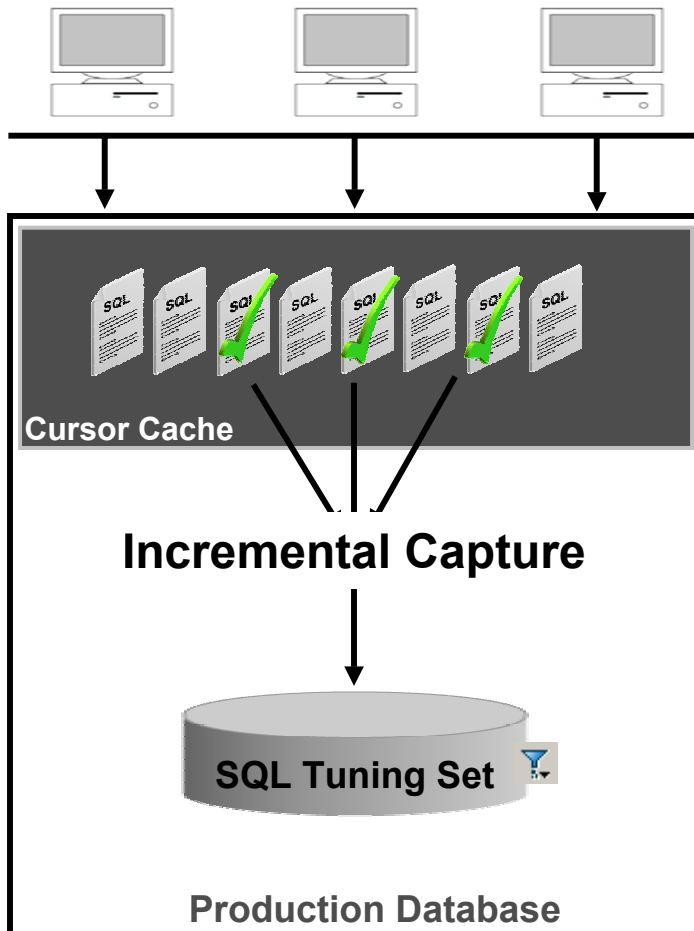
Capture  
(SQL\*Trace)

Copy to  
Test

Remote Execute  
SQL

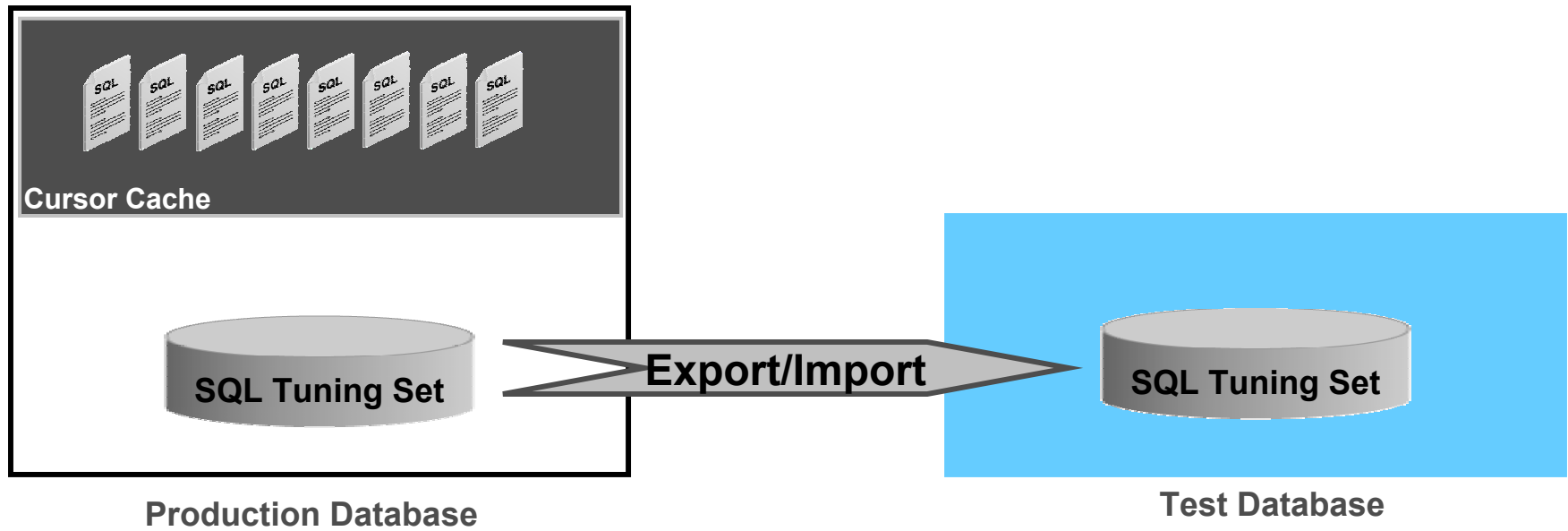
Compare  
Performance

# Step 1: Capture SQL Workload



- Capture workload using
  - SQL\*Trace (Oracle 9i or 10.1)
  - SQL Tuning Set (Oracle 10.2 or 11.1)
- Covert SQL\*Trace workload into STS (SQL Tuning Set)
- Incremental capture used to populate STS from cursor cache over a time period
- STS includes:
  - SQL Text
  - Bind variables
  - Execution plans
  - Execution statistics
- STS's filtering and ranking capabilities filters out undesirable SQL

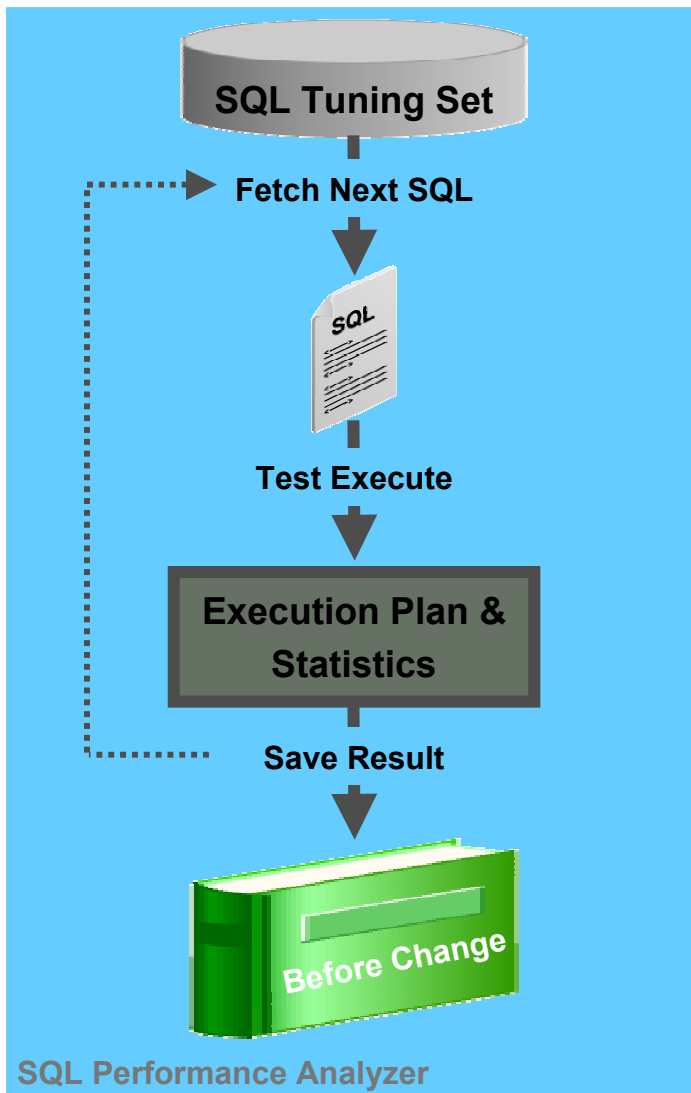
## Step 2: Move SQL Workload to Test System



- Copy SQL\*Trace file(s) to test system and convert into STS
- Copy STS to staging table ("pack")
- Transport staging table to test system (datapump, db link, etc.)
- Copy STS from staging table ("unpack")

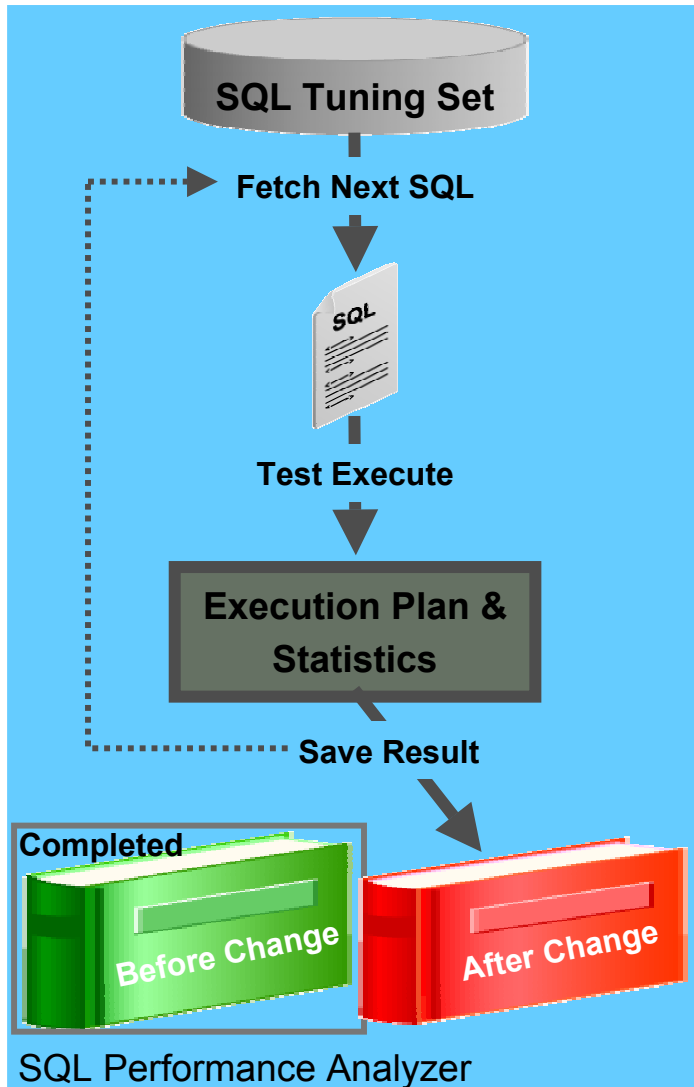


# Step 3: Execute SQL Before Making Change



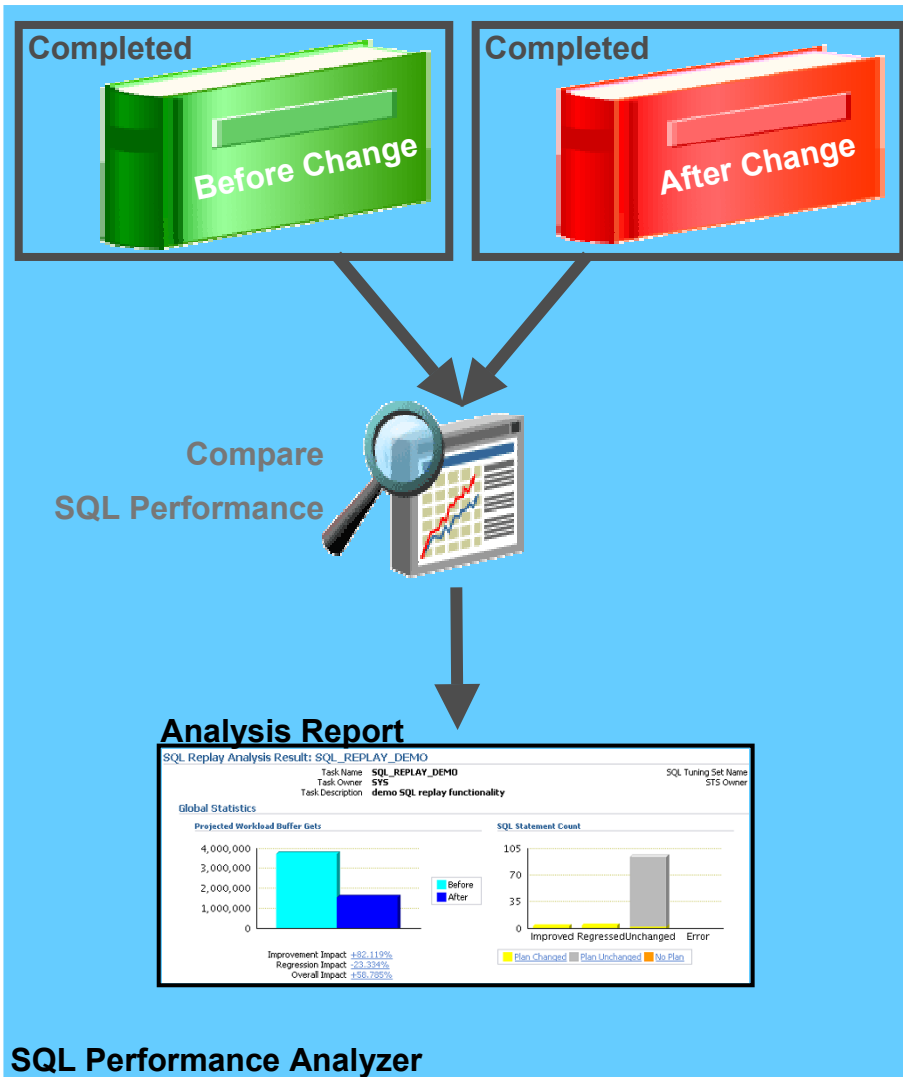
- Establishes SQL workload performance baseline
- SQL execution plan and statistics captured
- SQL executed serially (no concurrency)
- Each SQL executed only once
- DDL/DML skipped
- Option to do Explain Plan only analysis
- SQL\*Trace capture does not require this step as the trace file has the necessary execution stats

# Step 4: Execute SQL After Making Change



- **Manually implement the planned change**
  - Database upgrade, patches
  - Optimizer statistics refresh
  - Schema changes
  - Database parameter changes
  - Tuning actions, e.g., SQL Profile creation
- **Re-execute SQL after change**
  - Gathers new SQL execution plans and statistics

# Step 5: Compare & Analyze Performance



- Compare performance using different metrics, e.g.,
  - Elapsed Time
  - CPU Time
  - Optimizer Cost
  - Buffer Gets
- SPA Report shows impact of change for each SQL
  - Improved SQL
  - Regressed SQL
  - Unchanged SQL
- Fix regressed SQL using SQL Tuning Advisor or SQL Plan Baselines

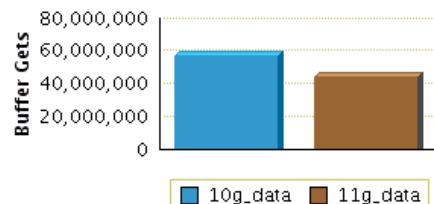
# SPA Report

## SQL Performance Analyzer Task Result: SYS.UPGRADE\_10G11G

Task Name	UPGRADE_10G11G	SQL Tuning Set Name	OOW 54G	Replay Trial 1	10g_data
Task Owner	SYS	STS Owner	SYS	Replay Trial 2	11g_data
Task Description	test upgrade to 11g	Total SQL Statements	54	Comparison Metric	Buffer Gets
		SQL Statements With Errors	0		

### Global Statistics

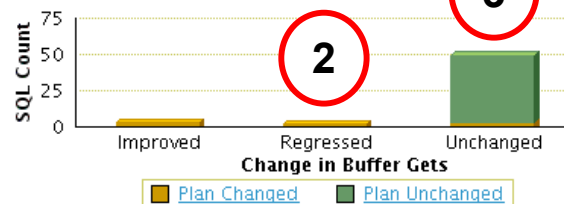
#### Projected Workload Buffer Gets



Improvement Impact **24%** ↑  
Regression Impact **-2%** ↓

Overall Impact **22%** ↑

#### SQL Statement Count



Plan Changed Plan Unchanged

#### Recommendations

Oracle offers two options to fix regressed SQL resulting from plan changes:

Use the better execution plan from SQL Trial 1 by creating SQL Plan Baselines.

Create SQL Plan Baselines

Explore alternate execution plans using SQL Tuning Advisor.

Run SQL Tuning Advisor

### Top 10 SQL Statements Based on Impact on Workload

SQL ID	Net Impact on Workload (%)	Buffer Gets		Net Impact on SQL (%)	% of Workload		Plan Changed
		10g_data	11g_data		10g_data	11g_data	
↑ g4dzf4ak4rus2	12.000	20,318,458.000	13,502,097.000	33.550	35.780	30.670	Y
↑ gfacm5jr3rz9j	11.990	6,990,541.000	180,401.000	97.420	12.310	0.410	Y
↓ 2ny751aat2vd9	-0.820	12,973,052.000	13,440,825.000	-3.610	22.850	30.530	Y
↓ c2fb0ug5p7d4p	-0.750	12,740,524.000	13,165,998.000	-3.340	22.440	29.910	Y
↑ 2wtgxbjz6u2by	0.050	244,678.000	218,533.000	10.690	0.430	0.500	Y

# SPA Report

## Regressed SQL Statements

### Regressed SQL Statements

SQL ID	Net Impact on Workload (%)	Buffer Gets		Net Impact on SQL (%)	% of Workload		Plan Changed
		10g_data	11g_data		10g_data	11g_data	
2ny751aat2vd9	-0.820	12,973,052.000	13,440,825.000	-3.610	22.850	30.530	Y
c2fb0u...	-0.750	12,					

### SQL Details: 2ny751aat2vd9

Parsing Schema DWH\_TEST

Execution Frequency 1

[Schedule SQL Tuning Advisor](#)

#### SQL Text

#### Single Execution Statistics

Execution Statistic Name	Net Impact on Workload (%)	Execution Statistic Collected		Net Impact on SQL (%)	% of Workload	
		10g_data	11g_data		10g_data	11g_data
Elapsed Time	-4.340	70.518	89.593	-27.050	16.060	24.170
Parse Time	-13.830	0.207	0.312	-50.720	27.270	32.470
CPU Time	-5.700	64.704	85.188	-31.660	18.010	24.200
Buffer Gets	-0.820	12,973,052.000	13,440,825.000	-3.610	22.850	30.530
Optimizer Cost	0.170	982.000	658.000	32.990	0.530	0.360
Disk Reads	10.800	7,011.000	5.000	99.930	10.810	1.850
Direct Writes	10.950	6,968.000	0.000	100.000	10.950	0.000
Rows Processed	0.000	111.000	111.000	0.000	0.000	0.000

### Plan Comparison

#### 10g\_data

Plan Hash Value 393503022

[Expand All](#) | [Collapse All](#)

Operation	Line ID	Object	Rows	Cost
SELECT STATEMENT	0		1	967
HASH	1		1	967
TABLE ACCESS	2	FACT_PD_OUT_ITM_293	1	966
NESTED LOOPS	3		1	966
MERGE JOIN	4		1	320
SORT	5		90	315
TABLE ACCESS	6	ADM_PG_FEATUREVALUE	1	2
NESTED LOOPS	7		90	314

# Use Cases



# Database Upgrade: 9.2/10.1\* → 10.2

## **Scenario 1:**

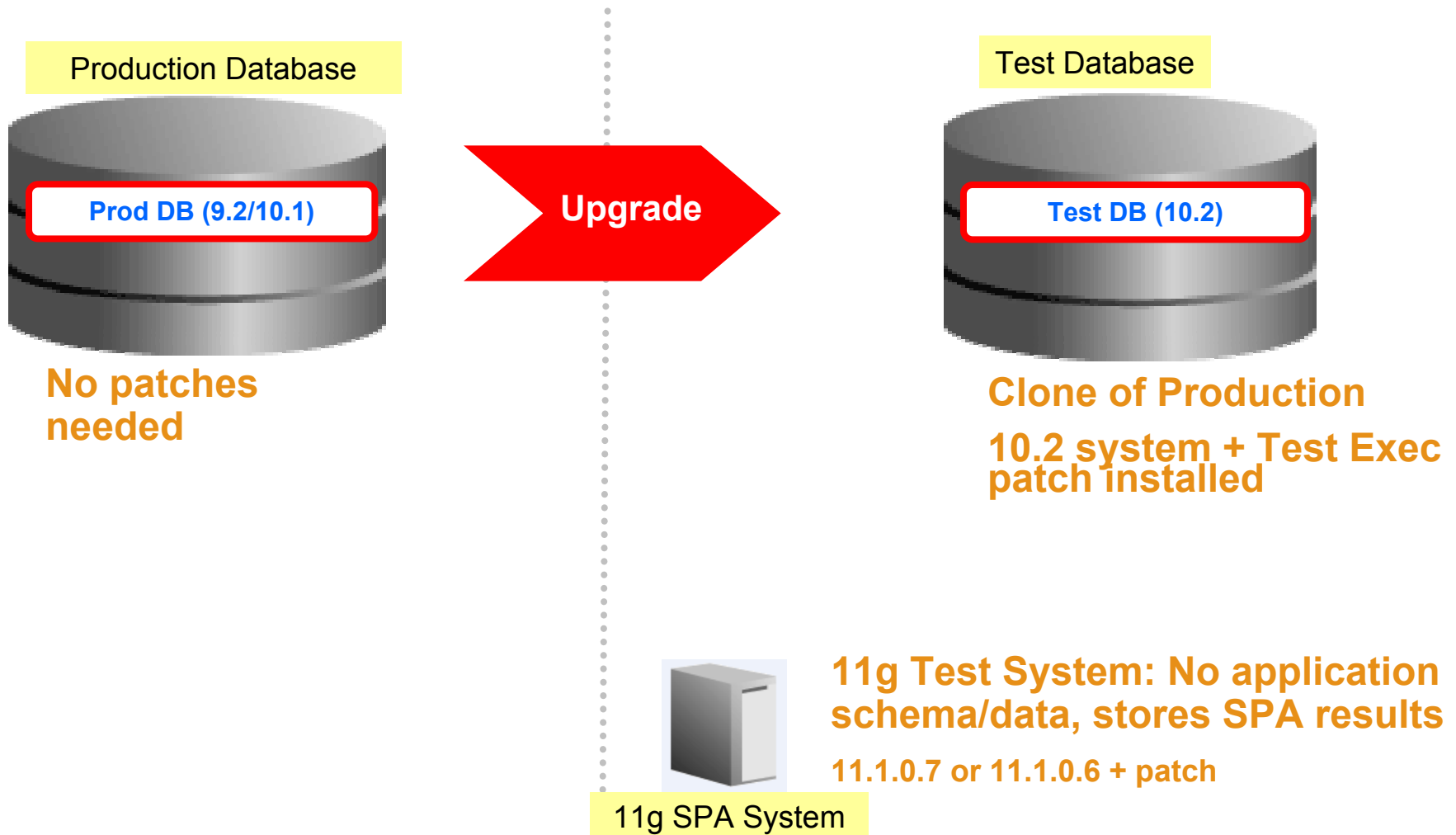
I have heard premier DB support for 9.2 has ended, I want to upgrade 10.2 database release. How can I use 11g SPA functionality to accomplish the upgrade?

## **Goal:**

Assess impact of upgrade on SQL workload performance on a test system using SPA so that there are no surprises after upgrade.

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

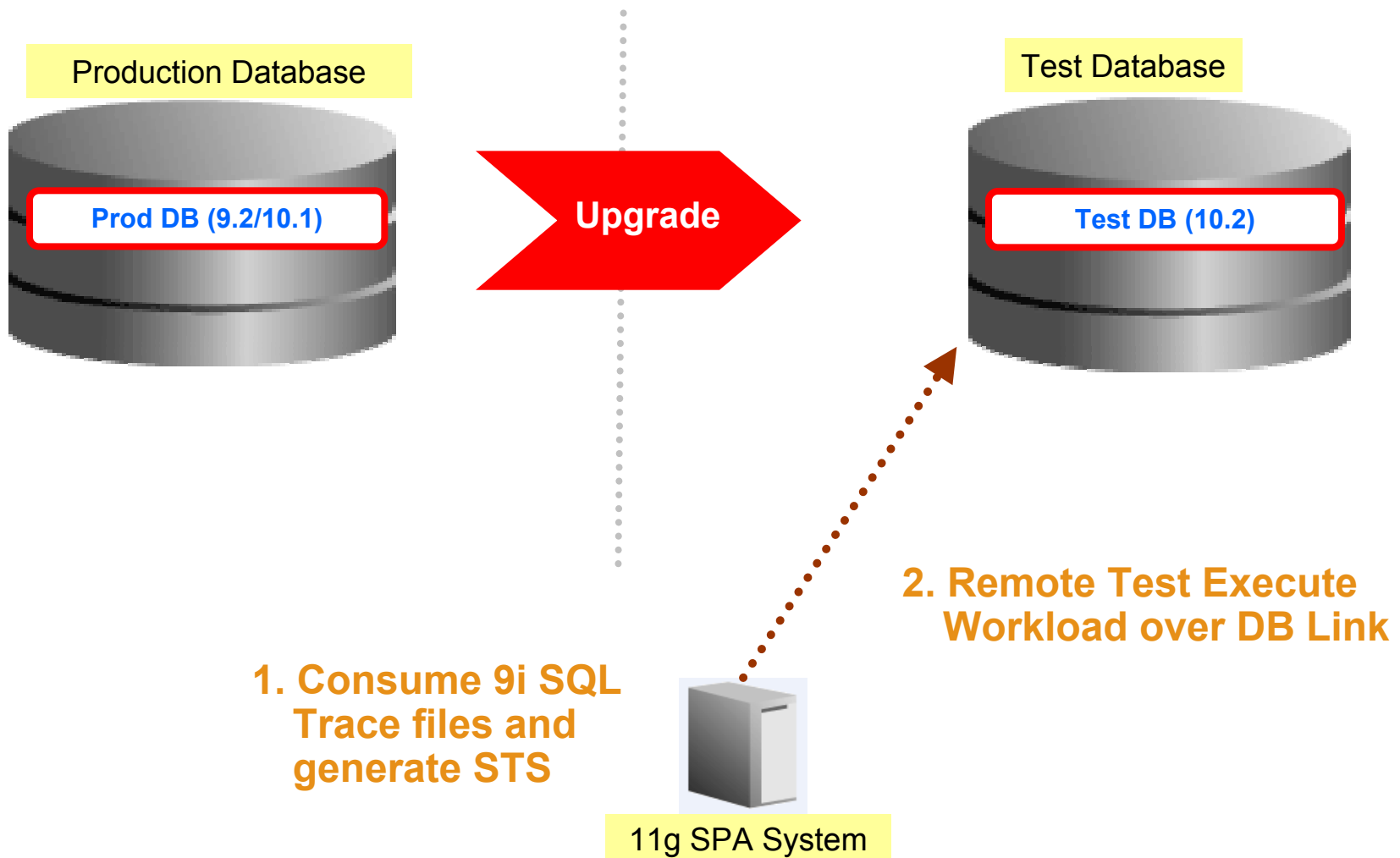
## System Setup



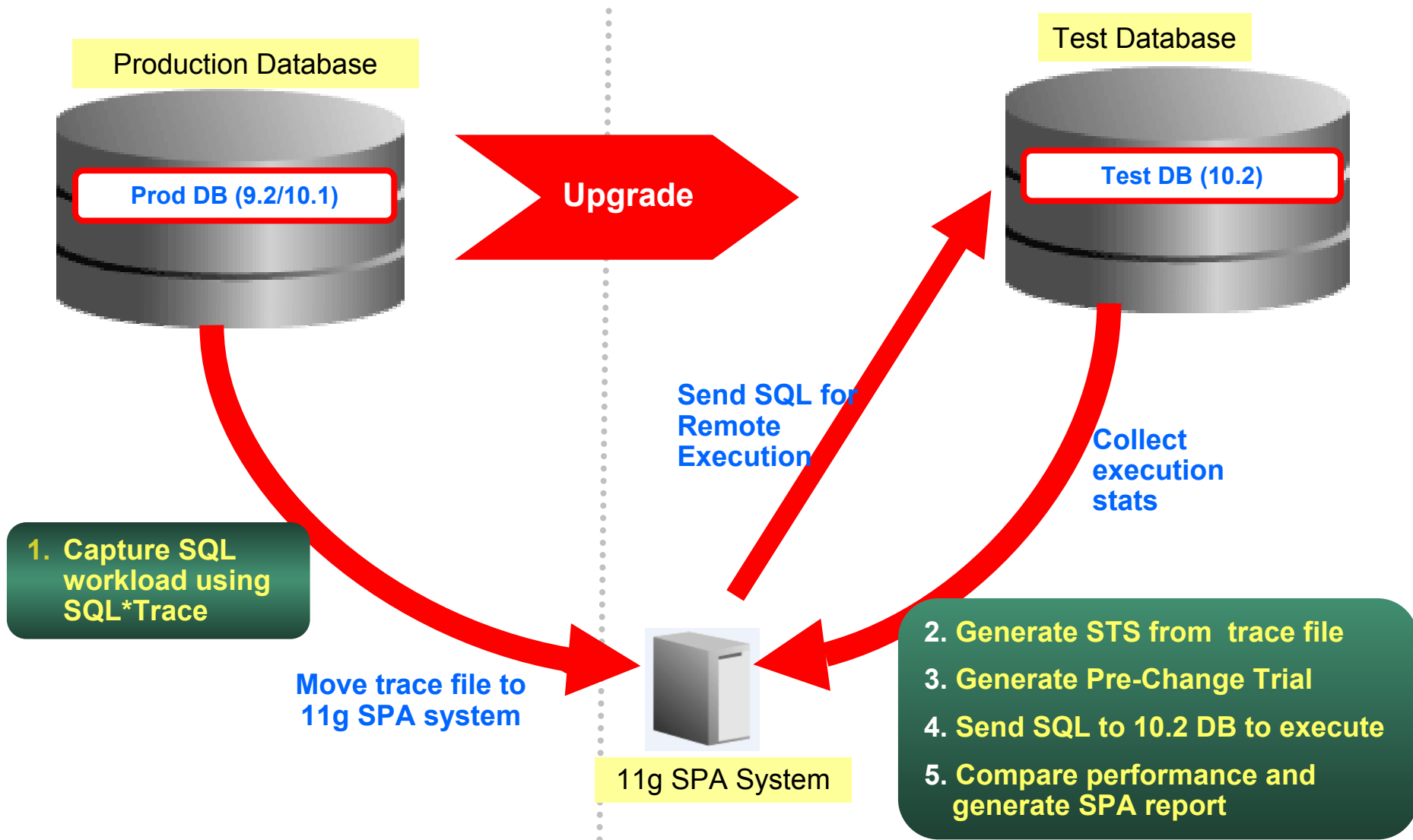


# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

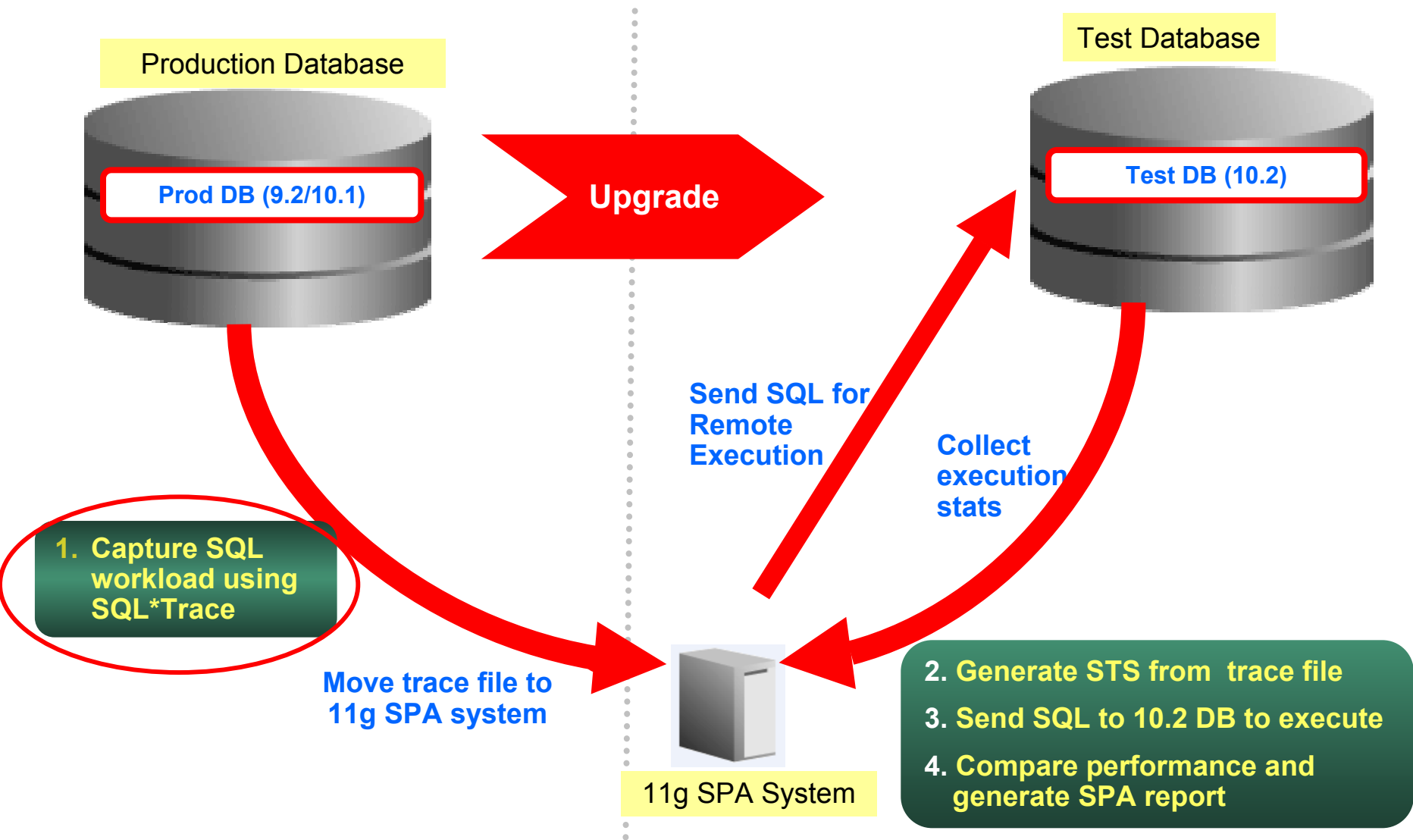
## SPA Enhancements



# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2



# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2



# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

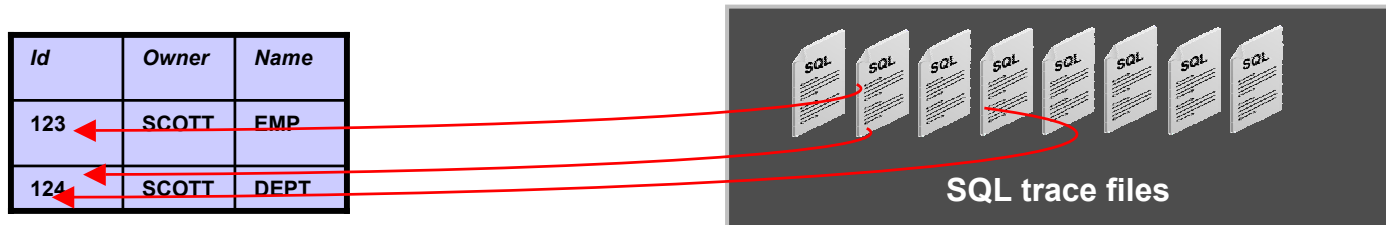
## Step 1: Capture SQL workload using SQL Trace

- Identify all interesting workloads such as month-end, daily peak, etc.
- Capture SQL trace for the workload, few sessions at a time
- Use dbms\_support/dbms\_monitor package, these support
  - bind value capture
  - tracing other running sessions
- SQL trace considerations
  - time\_statistics=true: Important for performance data
  - user\_dump\_dest
  - max\_dump\_file\_size
  - trace\_file\_identifier
  - Performance overhead: 10-15% for traced sessions

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

## Step 1: Capture SQL workload using SQL Trace (contd.)

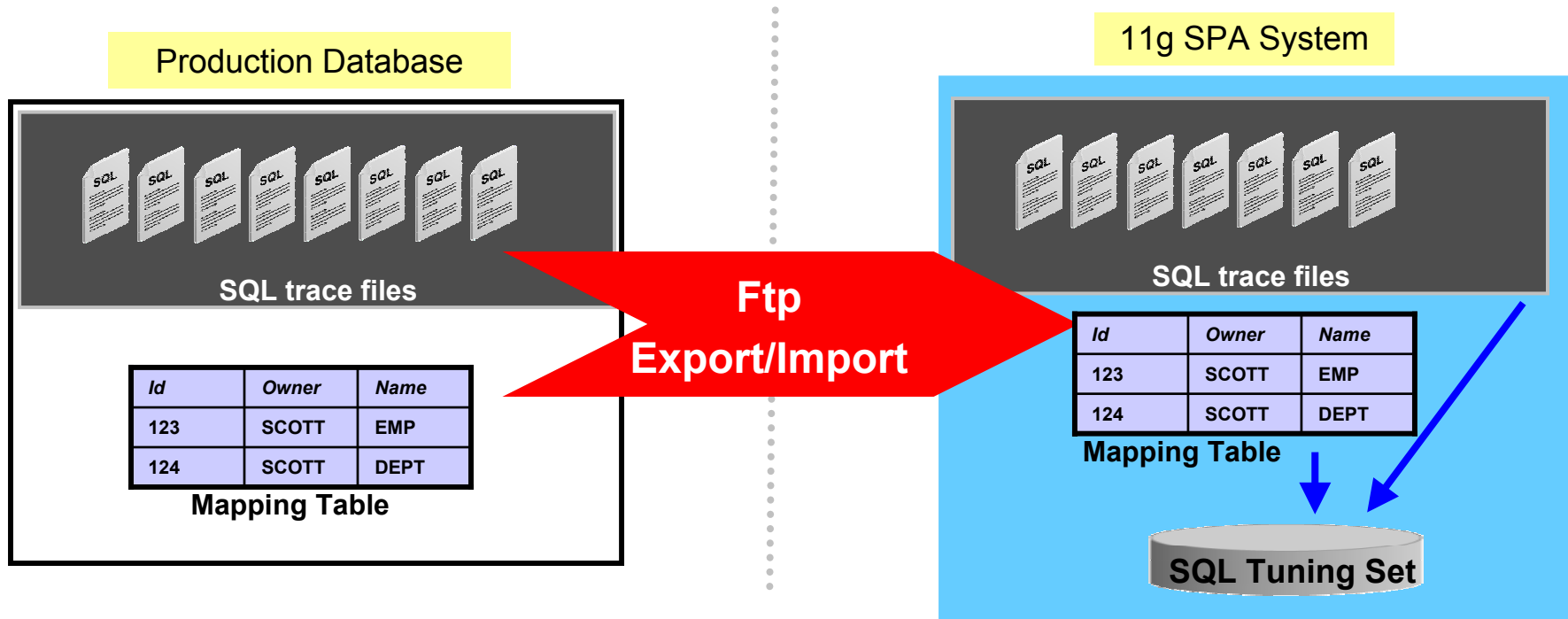
- SQL Trace files only have object identifiers
- Create mapping table to map object identifiers in trace files to schema names



\* SQL in Note pages/OTN

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

## Step 2: Transport Workload and Create STS



- Transport SQL trace files, mapping table using ftp/expdp/impdp..., etc.
- Create STS from trace files using dbms\_sqltune API (SQL in Notes page/OTN)
  - Specify directory object containing trace files, mapping table, STS name as input

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2 Step 3a: Create SPA Task

- Create SPA task on 11g SPA System

Database Instance: v11gk > Advisor Central > SQL Performance Analyzer > Logged in As SYS






**Guided Workflow**

Page Refreshed Sep 12, 2008 3:47:34 PM PDT  [View Data](#)

Real Time: 15 Second Refresh ▾

The following guided workflow contains the sequence of steps necessary to execute a successful two-trial SQL Performance Analyzer test.

Note: Be sure that the Trial environment matches the tests you want to conduct.

Step	Description	Executed	Status	Execute
1	Create SQL Performance Analyzer Task based on SQL Tuning Set		■	
2	Replay SQL Tuning Set in Initial Environment		■	
3	Replay SQL Tuning Set in Changed Environment		■	
4	Compare Step 2 and Step 3		■	
5	View Trial Comparison Report		■	

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2 Step 3a: Create SPA Task

Database Instance: v11gk > Advisor Central > SQL Performance Analyzer > Guided Workflow >

Logged in As SYS

## Create SQL Performance Analyzer Task

The SQL Performance Analyzer Task is a container for the execution of trial experiments designed to test the effects of changes in execution environment on the SQL performance of an STS.

\* Name

Owner **SYS**

Description   
**TIP** Use the description to characterize the intended SQL Performance Analyzer investigations.

## SQL Tuning Set

The SQL Tuning Set is the basis for SQL Performance Analyzer Task experiments. The STS should represent a coherent set of SQL for the changes being investigated (e.g. full workload for an upgrade test).

\* Name

**TIP** You can create a new STS here: [Link to STS Creation Wizard](#)

Cancel

Create



## Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

### Step 3b: Establish “Before Change” Trial

- Trial Creation Method: Select “Build From SQL Tuning Set” option to use plans and statistics from 9i

Database Instance: v11gk > Advisor Central > SQL Performance Analyzer > Guided Workflow:

### Create SQL Trial

SQL Trials capture execution performance of the SQL Tuning Set under given optimizer environment.

SQL Performance Analyzer Task **SYS.UPGRADE\_TEST**  
SQL Tuning Set **APPS.HR\_WORKLOAD**

\* SQL Trial Name

SQL Trial Description

Creation Method

**Schedule**

Time Zone

☒ Immediately

- Execute SQLs Locally
- Execute SQLs Remotely
- Generate Plans Locally
- Generate Plans Remotely
- Build From SQL Tuning Set**

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

## Step 4: Establish “After Change” Trial

### Guided Workflow: SYS.UPGRADE TEST

#### Create SQL Trial

SQL Trials capture execution performance of the SQL Tuning Set under a given optimizer environment.

SQL Performance Analyzer Task **SYS.T2**

SQL Tuning Set **APPS.HR\_WORKLOAD**

\* SQL Trial Name


SQL Trial Description

Creation Method

Per-SQL Time Limit

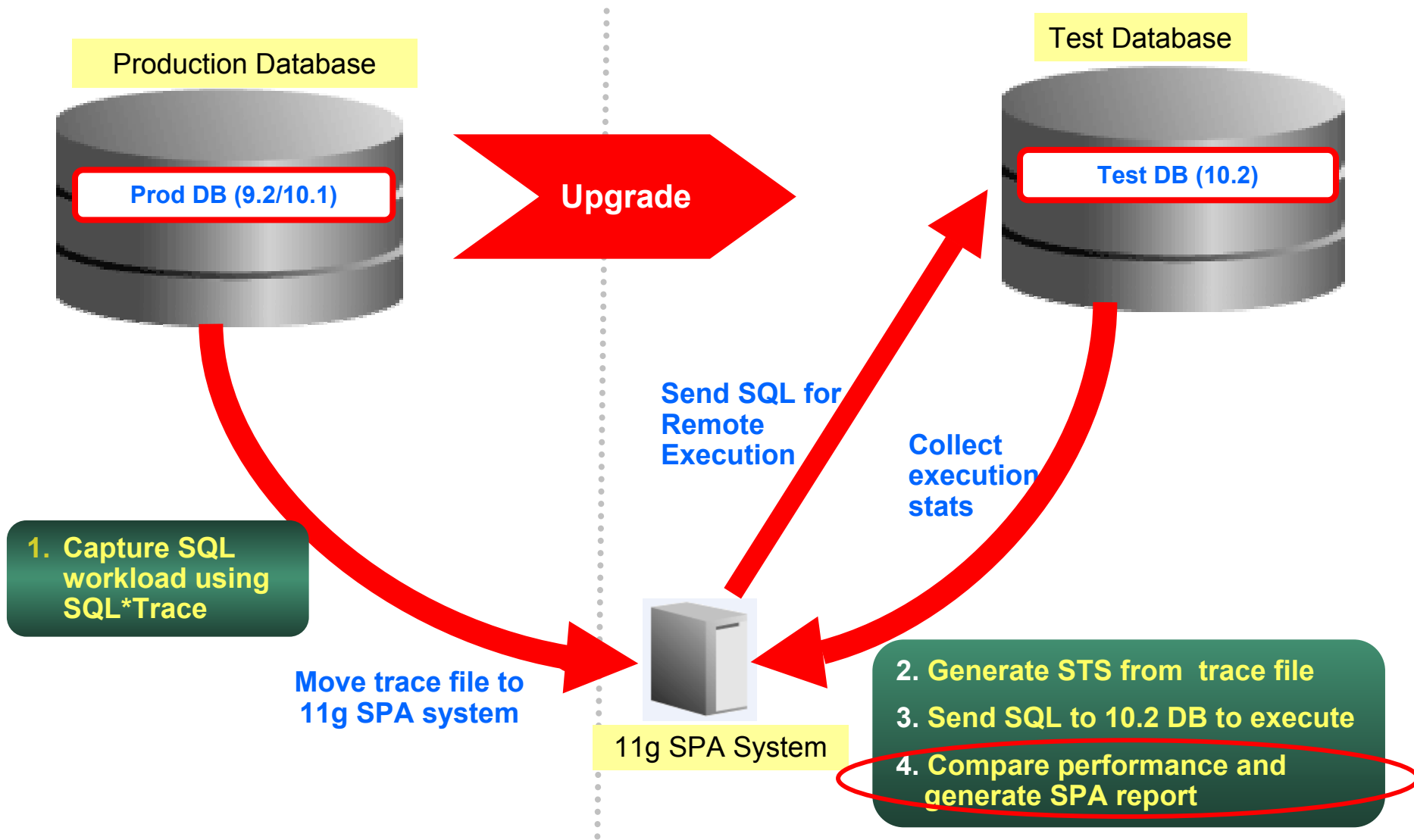
✓ **TIP** Time limit is on elapsed time of test execution of SQL.

\* Database Link



✓ **TIP** Provide a PUBLIC database link connecting to a remote user with privileges to execute the Tuning Set SQL.

# Scenario 1: Database Upgrade: 9.2/10.1 → 10.2



## Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

### Step 5: Compare and Generate Report

- Compare Pre-Change and After-Change Trials based on a performance metric
  - Oracle recommends using CPU\_TIME and BUFFER\_GETS metrics
  - Use multiple metrics that provide repeatable and comprehensive statistics

# Fixing Regressed SQL

- Systematic problems
  - Check un-analyzed tables, PGA memory, statistics collection, system statistics
  - Refer “Upgrading from Oracle 9i to 10g: What to expect from the Optimizer” on OTN
- For statements suffering from isolated problems use one of the following fixes
  - SQL Profiles: Implement Profiles recommended by SQL Tuning Advisor (STA)
  - Stored Outlines\*\*: If no profile was recommended by STA, then capture Stored Outlines in 9i for the targeted SQL statements. Import stored outline into 10g.

# Database Upgrade: 10.2.0.x → 10.2.0.y

## **Scenario 2:**

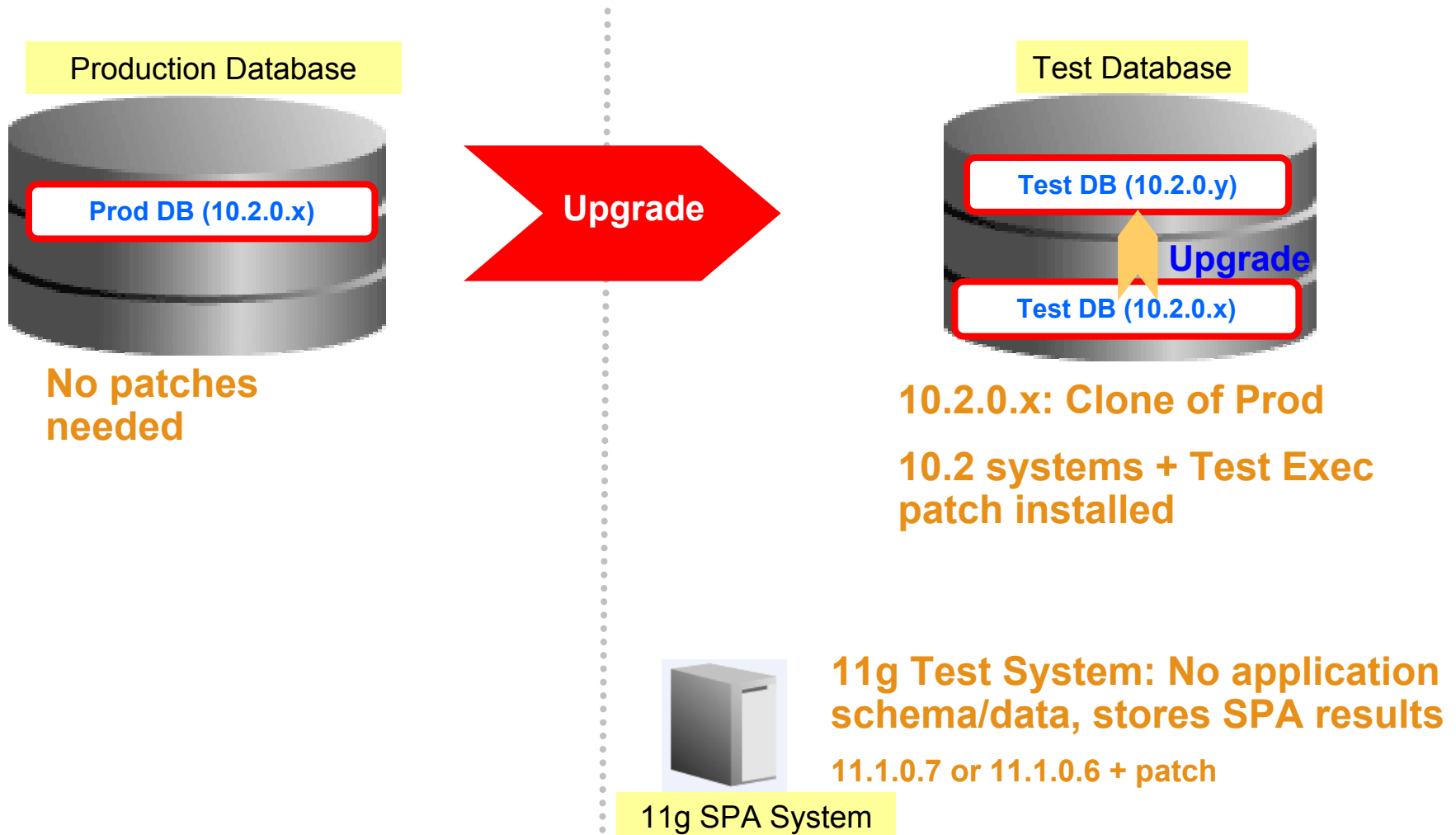
One of the database I'm managing is on 10.2.0.2. How can I use 11g SPA functionality to accomplish an 10.2.0.4 patchset upgrade?

## **Goal:**

Assess impact of upgrade on SQL workload performance using SPA so that there are no surprises after upgrade.

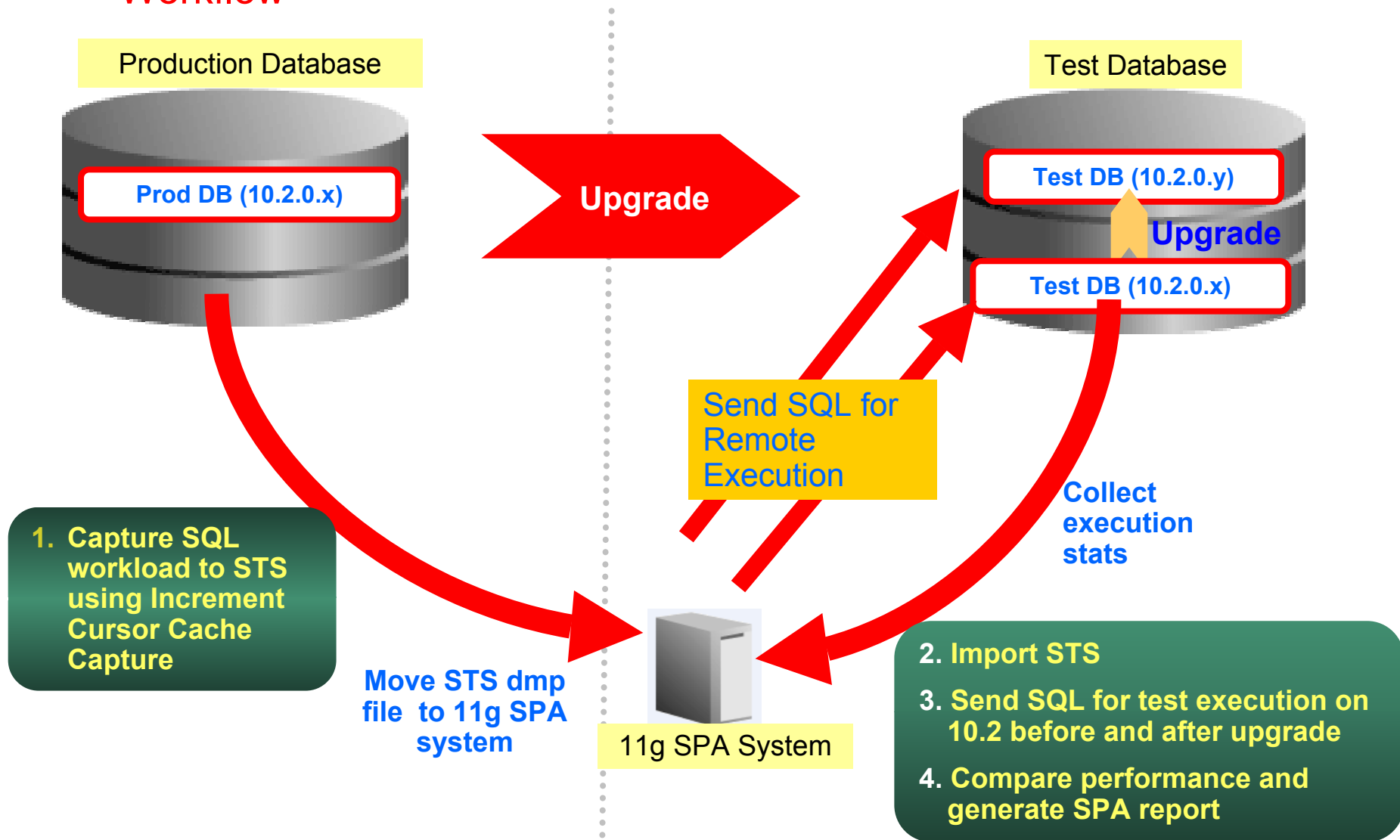
## Scenario 2: Database Upgrade: 10.2.0.x → 10.2.0.y

### System Setup



# Scenario 1: Database Upgrade: 10.2.0.x → 10.2.0.y

## Workflow





## Scenario 3: Using SPA Functionality for 9i/10g → 11g Upgrades

- Similar workflow as Scenario 2
- Use 11g SPA system and test execute on 10g/11g source and destination target databases
  - Stores results of experiments separately
  - Allows use of latest releases for 11g SPA system

# Evaluating Optimizer Statistics Refresh

## **Scenario 4:**

Can I use SPA to check if any SQL statements regressed due to optimizer statistics refresh on my 10.2 production databases. If so, how can I evaluate the refreshed optimizer statistics?

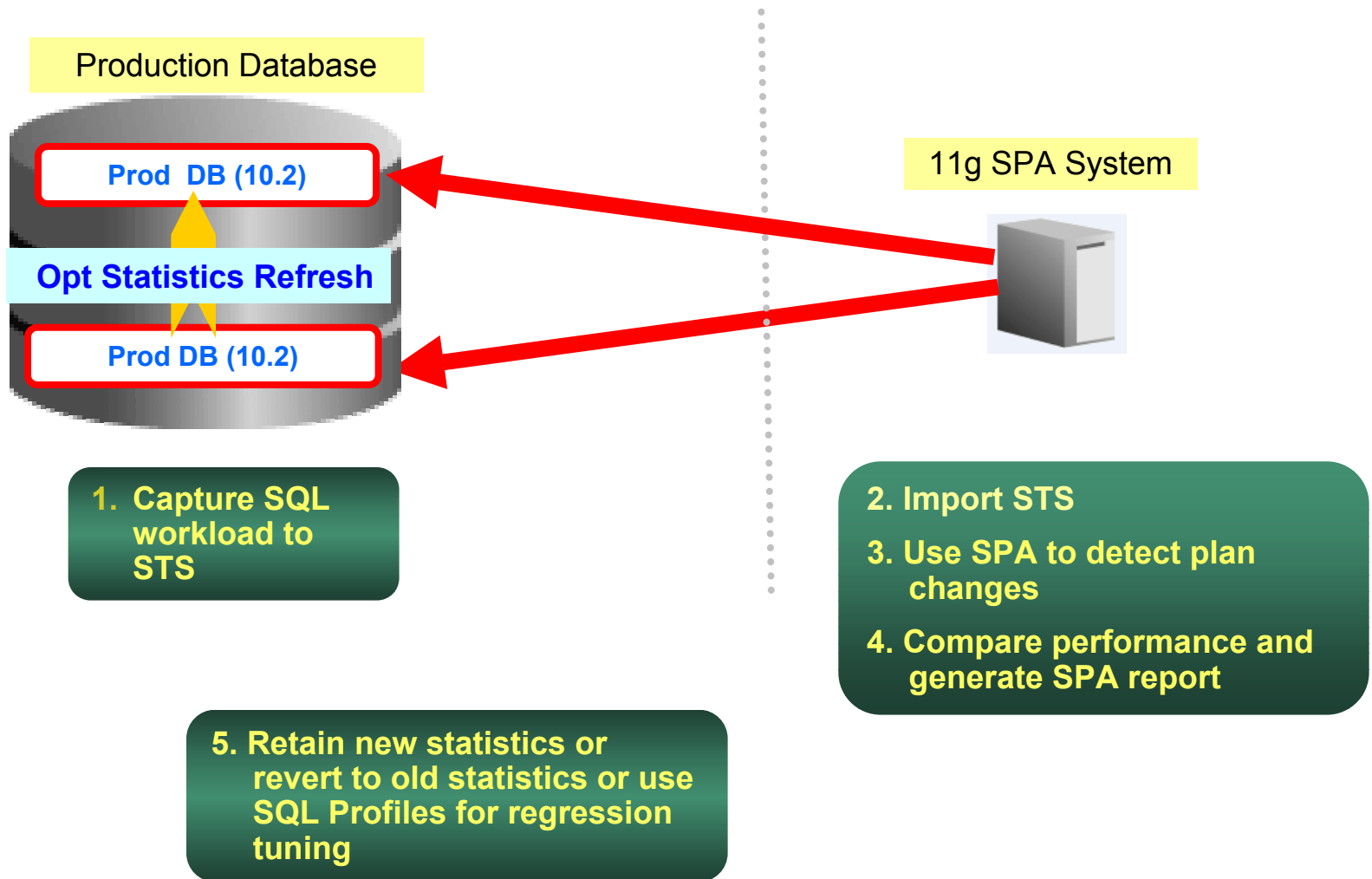
## **Goal:**

Assess impact of optimizer statistics gathering on SQL workload performance on production system & make sure are no negative effects of the change

# Evaluating Optimizer Statistics Refresh

- Assumptions
  - Optimizer has already gathered statistics on the database
  - Statistics refreshed periodically
  - No prod copy is available on test
- Use “11g SPA system” to evaluate optimizer statistics on 10.2 production database
  - Remote test execute before/after statistics refresh
- Analyze SPA report and take appropriate action
  - Overall improvement but few SQL regressions
    - Solution: Use SQL Profiles for regressed SQL
  - No improvement and many regressions
    - Solution: Revert to old statistics: Use optimizer statistics retention/history feature
  - For Oracle Database 11g, use publish pending statistics feature to publish statistics after evaluation of statistics

# Evaluating Optimizer Statistics Refresh for 10.2



# Real-World Deployments



# Large International Hotel Chain

## Challenge

- Upgrade critical customer-facing application providing rates for room reservations from Oracle Database 10.2.0.4 to 11.1
- Highly volatile data where plan stability is critical
- Unsuccessfully used synthetic queries to test previous upgrades

## Solution Approach

- SQL Performance Analyzer to identify SQL regressions
- SQL Profiles to tune SQL transparently
- SQL Plan Baselines for plan stability

## Benefit

- Very successful upgrade. No surprises!
- Predictable performance and SLAs
- Reduced testing time from 5 months to 10 days

# E-Business Suite (EBS) Certification and Testing

## Challenge

- Certify EBS release 11i, R12 against Oracle Database 11g
- Complex & large workload: More than **650K** unique SQL statements need to be validated
- Ensure application optimized for Oracle Database 11g
- Difficult to perform realistic and efficient testing with previous (home-grown) tools

## Solution Approach

- SQL Performance Analyzer to run regression tests and identify performance deviations
- Regressions reported to base development for fixes

## Benefit

- Reduced testing time from 21 to 2 days for each release
- Faster and higher quality testing
- Faster adoption and certification of newer features

*O* & *A*



# Real Application Testing Applicable for Pre-11g Database Releases

Feature	Capture From	Test Changes In
SQL Performance Analyzer	9i R2	10g R2 or 11g
	10g R1	10g R2 or 11g
	10g R2	10g R2 or 11g
Database Replay	9i R2	11g
	10g R2	11g

- **SQL Performance Analyzer**
  - Capture on 9i, 10.1, 10.2 database releases
  - Test changes in 10.2 & above
- **Database Replay**
  - Capture on 9i, 10.2 database releases
  - Test changes in 11.1 & above

