Insert Information Protection Policy Classification from Slide 12
Step-by-Step Cookbook for Identifying and Tuning SQL Problems

Ashish Agrawal - Consulting Product Manager, Oracle

Baki Şahin - Database Operation Supervisor, AveA Turkey
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.
Program Agenda

- Why SQL statements regress?
- Identifying problematic SQL
- Tuning SQL
- Preventing SQL problems
- Real-World Customer Experiences
  - AveA, Turkey
  - S. Corporation, Korea
Program Agenda

- Why SQL statements regress?
- Identifying problematic SQL
- Tuning SQL
- Preventing SQL problems
- Real-World Customer Experiences
  - AveA, Turkey
  - S. Corporation, Korea
Why SQL statements regress?

3 Broad Categories

- Optimizer
- Application
- Resource and contention Issues
Why SQL statements regress?

Optimizer-related

- Stale/Missing statistics
- Overly general statistics or incorrect histograms
- Improper optimizer configuration
- Upgraded database: new optimizer
- Changing statistics (refresh)
- Changing data (plans do not scale with data)
- Bind-sensitive SQL with bind peeking
- Not parallelized (no scaling to large data)
- Improperly parallelized (skews, RAC, etc.)
Why SQL statements regress?

Application-related

- Missing access structures
- Poorly written SQL statements
- Literal usage

Bad execution plans
- Full table scans
- Cartesian join

Poorly written SQL statements
- Convert literal once and not the whole column
Why SQL statements regress?

Resource and Contention Issues-related

Hardware resource crunch
- CPU, Memory, IO, Network

Data fragmentation

Logical Contention
- Row lock contention
- Block update contention

Example:-
enq: TX - allocate ITL entry
enq: TX - contention
enq: TX - index contention
enq: US - contention
Sub-optimal SQL performance: Symptoms

Symptoms

- Consumes high CPU, buffer gets, I/O, PGA memory
  - Long running SQL or significantly different runtimes
  - High I/O, CPU, memory, network waits
  - TX Enqueue Waits, Row Lock Contention
  - Plan regression
  - SQL appear in
    - Top Activity Page
    - ASH Analytics Page
    - ADDM Report, AWR Report, ASH reports
Program Agenda

- Why SQL statements regress?
- Identifying problematic SQL
- Tuning SQL
- Preventing SQL problems
- Real-World Customer Experiences
  - AveA, Turkey
  - S. Corporation, Korea
How to identify these SQL performance problems?

Identify SQL performance problems using:

- SQLs consuming high DB time
  - ADDM and ASH Analytics

- Long running SQLs and operations
  - Real-time SQL Monitoring & Database Operations Monitoring

- SQLs with execution plan changes
  - SQL Performance Analyzer (proactive)
Identify expensive SQL (Excessive DB time):
ADDM and ASH Analytics

ADDM
• Analyze current database performance through ADDM runs

ASH Analytics
• Next generation Top Activity Page
Identify expensive SQL: ADDM
SQL consuming too much DB time

ADDM

Throughput centric: Goal is to reduce ‘DB time’

Identifies top SQL

Shows SQL impact

Frequency of occurrence

Pinpoints root cause
Identify expensive SQL: ASH Analytics

Flexible Time Picker
Flexible Activity Chart
Flexible Top Chart
Flexible Top Chart

SQL ID
Top Level SQL ID
SQL Force Matching Signature
SQL Plan Hash Value
SQL Plan Operation
SQL Plan Operation Line
SQL Groups
Top Level SQL Groups

User Session
Activity (Average Active Sessions)
ASH Analytics
Active Reports Demo
Identify expensive SQL: ASH Analytics

Active Reports Demo

Click on Save
Identify long running SQL:
Real-Time SQL Monitoring

Automatically monitors instances of long running SQL, PL/SQL executions

Enabled out-of-the-box with no performance overhead

Obviates need to trace individual SQL

Shows global PL/SQL and SQL level statistics

Guides tuning efforts

Monitored SQL Executions

<table>
<thead>
<tr>
<th>S.</th>
<th>Duration</th>
<th>Type</th>
<th>ID</th>
<th>SQL Plan Hash</th>
<th>User</th>
<th>P.</th>
<th>Database Time</th>
<th>IO Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42.0s</td>
<td>SQL</td>
<td>6hc35b1m03m9z</td>
<td>159</td>
<td>SH</td>
<td></td>
<td>5.7s</td>
<td>3,739</td>
</tr>
<tr>
<td>2</td>
<td>4.0s</td>
<td>SQL</td>
<td>f58fb5n0yvr7c</td>
<td>147</td>
<td>DBSNMP</td>
<td>4.5s</td>
<td>1,423</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>33.0s</td>
<td>SQL</td>
<td>fh8upax5cxsz</td>
<td>308</td>
<td></td>
<td></td>
<td>33.6s</td>
<td>752</td>
</tr>
<tr>
<td>4</td>
<td>15.0s</td>
<td>SQL</td>
<td>5k5207588w9ry</td>
<td></td>
<td></td>
<td>15.5s</td>
<td>543</td>
<td></td>
</tr>
</tbody>
</table>
Database monitoring of application jobs

- Grouping of SQLs, sessions for the application jobs
- Key scenarios: ETL operations, Quarter End Close jobs

Driven by application specified tagging

- Oracle Data Pump jobs automatically monitored
- Tagging ability in PL/SQL, OCI, JDBC

Visibility into top sql statements
Identify SQL plan changes: SPA

SPA

- Tests and predicts impact of system changes on SQL query performance
- Analyzes overall performance impact including improvements or regressions
- Common plan change scenarios
  - Database parameter, schema changes
  - Statistics gathering refresh
  - I/O subsystem changes, Exadata
  - Database upgrades or patches
Program Agenda

- Why SQL statements regress?
- Identifying problematic SQL
- Tuning SQL
- Preventing SQL problems
- Real-World Customer Experiences
  - AveA, Turkey
  - S. Corporation, Korea
Tuning SQL: SQL Tuning Advisor

- Multitenant database-aware
- All SQL’s are tuned across all PDB’s where it has executed before.
- Empowers the CDBA to tune across PDBs in one click
Tuning SQL: SQL Tuning Advisor

SQL Profiling

Automatic Tuning Optimizer verifies and adjusts its own cardinality estimates

Optimizer provides additional information to generate execution plan

Test executes the recommended execution plan for performance

Reviews execution history

✓ Ideal when cardinality estimates are wrong, collection of statistics on objects or creation of new indexes is required
Some optimizer related change in the environment results in a new plan being 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:
1) DBA can verify plan at any time.
2) Adaptive SPM, SPM Evolve Advisor runs daily in the scheduled maintenance window.

✓ Ideal when you need to preserve and use good, known and verified execution plans.
Tuning SQL: SQL Access Advisor

Recommendations

Indexes
- B-tree indexes
- Bitmap indexes
- Function-based indexes

Materialized views and view logs
- Fast refreshable
- Full refreshable MVs

Partition table
- Range, Interval, Hash, Range-Hash, Range-List type, List
- New partitioning schemes on already partitioned tables

Partition index
- Local, Range, Hash type

✓ Ideal when you need advice for creation of indexes, mviews and partition for your entire workload
Tuning SQL: Real-time SQL Monitoring Case Study

I enabled parallel query, yet this query is taking so long. What’s going on?

Parallel server downgrades?

• Uncontrolled parallel execution
• Parallel Server availability
• Object level settings
• Session level settings

✓ Ideal when you need to tune long running SQLs, complex queries with big execution plans, parallel queries, DML and DDL statements, Exadata smart scans, cases of a poor indexing strategies
Tuning SQL: Real-time SQL Monitoring

Insert executed with parallel hint
### Parallel Tab

<table>
<thead>
<tr>
<th>Parallel Server</th>
<th>Database Time</th>
<th>Wall Activity %</th>
<th>IO Requests</th>
<th>Buffer Gets</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Parallel Servers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Coordinator</td>
<td>8.0s</td>
<td></td>
<td>14K</td>
<td>1,750K</td>
</tr>
<tr>
<td>Parallel Set 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 1 (p000)</td>
<td>27.6s</td>
<td></td>
<td>1,712</td>
<td>324K</td>
</tr>
<tr>
<td>Parallel Server 2 (p001)</td>
<td>31.7s</td>
<td></td>
<td>1,797</td>
<td>336K</td>
</tr>
<tr>
<td>Parallel Server 3 (p002)</td>
<td>28.1s</td>
<td></td>
<td>1,721</td>
<td>324K</td>
</tr>
<tr>
<td>Parallel Server 4 (p003)</td>
<td>28.4s</td>
<td></td>
<td>1,728</td>
<td>335K</td>
</tr>
<tr>
<td>Parallel Server 5 (p004)</td>
<td>29.2s</td>
<td></td>
<td>1,716</td>
<td>325K</td>
</tr>
<tr>
<td>Parallel Server 6 (p005)</td>
<td>34.0s</td>
<td></td>
<td>1,770</td>
<td>333K</td>
</tr>
<tr>
<td>Parallel Server 7 (p006)</td>
<td>28.0s</td>
<td></td>
<td>1,763</td>
<td>335K</td>
</tr>
<tr>
<td>Parallel Server 8 (p007)</td>
<td>24.5s</td>
<td></td>
<td>1,700</td>
<td>324K</td>
</tr>
</tbody>
</table>

- Parallel Coordinator busy for the entire duration!!
Tuning SQL: Real-Time SQL Monitoring
Solution: Enabled Parallel DML

☑ Parallel Slaves busy for the entire duration!!!
Program Agenda

- Why SQL statements regress?
- Identifying problematic SQL
- Tuning SQL
- Preventing SQL problems
- Real-World Customer Experiences
  - AveA, Turkey
  - S. Corporation, Korea
Preventing SQL problems:
Finding Regressed SQL statements proactively using SPA
SPA Challenges

Running SPA on:

Test System: Safe But…
- Requires separate HW
- Data in test system should be same as production
- Lengthy, error-prone task

Production System: Easier but…
- Could be resource intensive and impact production performance
- Changes needs to be manually scoped to private session
- Could take a long time to finish
- No resource control by default
SPA Quick Check
New in EM 12c Database Plug-in 12.1.0.5

Supports routine production change use cases
• Optimizer Gather Statistics
• Init.ora parameter changes
• Index creation
• Support for DB Release 11g and above

Simplifies precise identification of plan regressions
• Uses pre-selected STS that represents workload to be tested
• Default SPA settings

Designed and optimized for production use
• Optimal Trial or Explain Plan Mode
• Per SQL Time Limit, Resource Consumer Group
• Limits testing scope to private session
SPA Quick Check

Optimal Trial Mode

Optimal (Hybrid): This is the recommended mode. It finds SQLs with plan changes first by generating plan, then test-executes SQL statements with plan changes.

Test Execute: Test-execute every SQL statement and collect its execution plans and execution statistics.

Explain Plan: Generate explain plan for every statement in the SQL workload.

Identifies subset SQL workload with plan changes first

Test-executes only SQLs with plan changes

Minimizes use of production resources dramatically

• In general, resource consumption reduced in the range of 10x or more
SPA Quick Check: Setup

Minimal use of production resources

- Per-SQL Time Limit - prevents any runaway SQLs
- Disables Multiple Executions
- Uses Resource Consumer Group
- Executes only the query part of the DML without modifying data
SPA Quick Check
Launch SPA in-line with Gather Optimizer Statistics workflow

**DBA Best Practice: Validate Optimizer Statistics Gathering**

**Validate With SQL Performance Analyzer**

- Validate impact of stats on SQL performance prior to publishing (recommended). The database global statistics gathering option PUBLISH will be set to FALSE temporarily during the process.
**SPA Quick Check**

**Workflow for validating Gather Optimizer Statistics**

Optimizer statistics gathering option PUBLISH set to FALSE temporarily during the process

Four trials automatically executed, compared and reports generated

<table>
<thead>
<tr>
<th>SQL Trial Name</th>
<th>Description</th>
<th>Created</th>
<th>SQL Executed</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST_TRIAL</td>
<td>Trial with current optimizer statistics.</td>
<td>8/22/13 11:58 AM</td>
<td>No</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>SECOND_TRIAL</td>
<td>Trial with pending optimizer statistics. Try to find SQLs with plan change.</td>
<td>8/22/13 11:58 AM</td>
<td>No</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>FOURTH_TRIAL</td>
<td>Trial with pending optimizer statistics, only for the SQLs with plan change.</td>
<td>8/22/13 11:58 AM</td>
<td>Yes</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>THIRD_TRIAL</td>
<td>Trial with current optimizer statistics, only for the SQLs with plan change.</td>
<td>8/22/13 11:58 AM</td>
<td>Yes</td>
<td>COMPLETED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trial Name 1</th>
<th>Trial Name 2</th>
<th>Compare Metric</th>
<th>Created</th>
<th>Status</th>
<th>Comparison Report</th>
<th>SQL Tune Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST_TRIAL</td>
<td>SECOND_TRIAL</td>
<td>Optimizer Cost</td>
<td>8/22/13 11:58 AM</td>
<td>COMPLETED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>THIRD_TRIAL</td>
<td>FOURTH_TRIAL</td>
<td>Elapsed Time</td>
<td>8/22/13 11:58 AM</td>
<td>COMPLETED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SPA Quick Check
Workflow for Validating Gather Optimizer Statistics

- Provides actionable recommendations on the report
- Publish the pending statistics
- Create SQL Tuning Set for regressed SQLs
- Offers two options to fix regressed SQL resulting from plan changes
- Create SQL Plan Baselines
- Explore alternate execution plans using SQL Tuning Advisor

Projected Workload Elapsed Time

<table>
<thead>
<tr>
<th>Elapsed Time (sec)</th>
<th>SQL Statement Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>0.03</td>
<td>1.0</td>
</tr>
<tr>
<td>0.06</td>
<td>1.5</td>
</tr>
<tr>
<td>0.09</td>
<td>Improved</td>
</tr>
<tr>
<td>0.09</td>
<td>Regressed</td>
</tr>
<tr>
<td>0.09</td>
<td>Unchanged</td>
</tr>
</tbody>
</table>

Change in Elapsed Time

- Improved
- Regressed
- Unchanged

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

1. Use the better execution plan from SQL Trial 1 by creating SQL Plan Baselines.
   - Create SQL Plan Baselines
2. Explore alternate execution plans using SQL Tuning Advisor.
   - Run SQL Tuning Advisor

Related Actions

- Publish new optimizer statistics.
- Publish Object Statistics

Create SQL tuning set for regressed SQLs.
- Create SQL Tuning Set
SPA Quick Check
Gather Optimizer Statistics – SPA Report

SPA Report without any regression
Preventing SQL problems: SPA Quick Check

Init.ora changes: Validate optimizer related init parameter change

First trial with current parameter values

Parameter change only effective in private session

Second trial with changed values

Generates comparison report

Provides actionable recommendations on the report (buttons)

Note: Applies to optimizer related changes and session parameters only
Preventing SQL problems: SPA Quick Check

New index creation: Use SPA Quick Check to find any plan changes

- First trial with current environment
- Create index in invisible mode
- Second trial with index invisible
- Generates comparison report

Provides actionable recommendations on the report (buttons)

Note: Indexes are only visible in a scoped manner to the session, uses optimizer_use_invisible_indexes=true in private session
Program Agenda

- Why SQL statements regress?
- Identifying problematic SQL
- Tuning SQL
- Preventing SQL problems
- Real-World Customer Experiences
  - AveA, Turkey
  - S. Corporation, Korea
11g Upgrade Project using Oracle Real Application Testing

Baki Şahin
Database Operation Supervisor
Avea is the youngest GSM operator of Turkey with its 13.6 million customers.

- The only GSM 1800 mobile operator of Turkey
- Founded in 2004 merged of 2 GSM Operators (Aria & Aycell)
- Nationwide customer base of 13.6 million
- Provide GSM service 98% of the population in Turkey
- Around 3000 people work for Avea
- Certified as an R&D company in 2010
Prepaid History Workload Description

Application

• All transaction related to prepaid customers except bonus
• Store last 6 months traffic
• Business rules for post-sales processes. (Rule engine)
• Configuration of subscriber services

Database

• Database size is ~2.5Tb
• Workload can roughly be classified into 2 parts:
  - OLTP transactions is done during business hours
  - Batch processing is done after business hours
Overview of Configuration

As is Configuration
Os: Sun OS 5.10
Software: 10.2.0.4
Hardware: Sparc M4000

To be Configuration
Os: Sun OS 5.10
Software: 11.2.0.3
Hardware: Sparc M4000
10.2 → 11g Database Upgrade

1. Capture SQL workload to STS
2. Transport STS

Production Database

Test Database

Test DB (11gR2)

Disaster DB (10.2)

Activate Data Guard & Upgrade

3. Establish 10.2 and 11g Trials
4. Compare performance and generate SPA report
5. Deploy Tuning and Change to Prod

Exp/imp or EM
Compare Performance

SQL Performance Analyzer Task Result: SYS.SPA

Overview

SQL Tuning Set Name: UPDATED_ST5
SQL Trial 1: pre_analyze
SQL Trial 2: post_analyze
Total SQL Statements: 91,267
Comparison Metric: Elapsed Time

Overall Result

Global Statistics | SQL Statements Breakdown

Projected Workload

Recommenations:
- The analysis suggests two options to improve SQL performance:
  - Use the lower execution plan from SQL Trial 1 by creating a SQL Plan Baseline.
  - Explore alternative execution plans using SQL Tuning Advisor.

Top 10 SQL Statements Based on Impact on Workload:

<table>
<thead>
<tr>
<th>SQL ID</th>
<th>Net Impact on Workload (%)</th>
<th>Elapsed Time (sec)</th>
<th>New Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SQL Trial 1</td>
<td>SQL Trial 2</td>
</tr>
<tr>
<td>3192045</td>
<td>-8.270</td>
<td>0.014</td>
<td>0.017</td>
</tr>
<tr>
<td>3192046</td>
<td>3.729</td>
<td>0.066</td>
<td>0.057</td>
</tr>
<tr>
<td>3192041</td>
<td>3.064</td>
<td>0.002</td>
<td>0.004</td>
</tr>
<tr>
<td>3192044</td>
<td>-3.274</td>
<td>0.008</td>
<td>0.010</td>
</tr>
<tr>
<td>3192047</td>
<td>1.700</td>
<td>0.010</td>
<td>0.012</td>
</tr>
<tr>
<td>3192048</td>
<td>-1.220</td>
<td>0.002</td>
<td>0.002</td>
</tr>
<tr>
<td>3192049</td>
<td>0.400</td>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>
Regressed SQL Statements

Plans unchanged

• Majority of plans unchanged
• Unlock statistics collections for tables
• Collects optimizer statistics again using 11g database
• Run SPA again

Plans changed – fixing regressions

• Create Baselines
• Work with developers
• Indexes created or dropped
• Run SPA again
Conclusion

Benefits for us

- Risk reduction
  - Replays All SQL
  - Real Bind Values
- Test production SQL workload before upgrade
- Tune regressed SQL statements
- No surprises when upgraded to 11.2.0.3
- Report what will happen before upgrade

- To be more safe
- Reusable process
- Reduced resource requirements
  - 1 team against 4 teams
- Fully proven recommendations
  - Easier to convince business
Agenda

1. Project Overview
2. Oracle Solution for Upgrade
3. Results
Project Overview

● Customer Information
  - S. Corporation: Manufacturing company in Korea.

● Target: ERP System
  - SAP ERP system.
  - ECC: Logistics, Finance, XI and other sub systems with RAC configuration
  - Production, Test, DR and etc.

● Necessity of 11g Upgrade
  - 11g upgrade for a stable service environment
  - 10g version support instability due to End of Support period
  - Database environment improvement through 11g new function utilization

● 11g Upgrade Schedule
  - Test Process: Procedure Test ▶ Stability Test ▶ Application Test (Function, Performance, Compatibility)
    ▶ Rehearsal ▶ Cut Over
## Solution for Upgrade

**Oracle Real Application Testing**

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Pain Point** | - Regression had to be checked before the upgrade due to business criticality  
- Time and cost issue for test and verification of more than one million SQLs within 4 months.  
- Low efficiency for SQL verification, if done manually. |
| **How**    | - PoC for more than 2 months to validate the real effectiveness of RAT and to assess potential upgrade risks, which can’t be found without RAT.  
- Sort sequence changing target program test through module source check.  
- Performance test & tuning for individual SQL through RAT SPA. (10g vs. 11g ) |
| **Result** | - Contribution to stable 11g Upgrade through SQL Change Risk exclusion.  
- 1 million SQLs Performance tuning through SQL Performance Analyzer utilization  
- Reduced upgrade project time through auto verification. |
## Results

<table>
<thead>
<tr>
<th>Factor</th>
<th>Content</th>
</tr>
</thead>
</table>
| **Database Performance Improvement** | ▪ Improved performance through the Optimizer enhancement : About 15~20% improvement  
▪ RAC efficiency, CPU usage monitoring improvement through the EM function improvements : About 20% improvement |
| **RAT Solution Utilization** | ▪ Over 1 million SQLs Performance testing using SPA  
▪ Stable 11g upgrade of the most mission critical system through SQL change risk elimination |
| **Collaboration**    | ▪ Verification exercises through the detailed procedure definition and sufficient tests.  
▪ Successful collaboration between the customer TFT and Oracle team (Local and Global) through close communication. |
# Database Manageability

## Recommended Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>General Session</th>
<th>Day</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>GEN8792</td>
<td>General Session: Database Management Innovations - Oracle Database 12c Manageability Highlights</td>
<td>Wednesday</td>
<td>10:15 AM</td>
<td>Moscone South – 103</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Session</th>
<th>Session</th>
<th>Day</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>CON9582</td>
<td>Oracle Exadata Management Deep Dive with Oracle Enterprise Manager 12c</td>
<td>Monday</td>
<td>12:15 PM</td>
<td>Westin - Metropolitan I</td>
</tr>
<tr>
<td>CON9573</td>
<td>Managing the Oracle Identity Management Platform with Oracle Enterprise Manager</td>
<td>Monday</td>
<td>1:45 PM</td>
<td>Moscone South - 130</td>
</tr>
<tr>
<td>CON8788</td>
<td>Maximizing Database Performance with Database Replay</td>
<td>Tuesday</td>
<td>10:30 AM</td>
<td>Moscone South - 308</td>
</tr>
<tr>
<td>CON9583</td>
<td>Harness the Power of Oracle Database 12c with Oracle Enterprise Manager Database as a Service</td>
<td>Tuesday</td>
<td>3:45 PM</td>
<td>Moscone South – 305</td>
</tr>
<tr>
<td>CON9579</td>
<td>Step-by-Step Cookbook for Identifying and Tuning SQL Problems</td>
<td>Wednesday</td>
<td>1:15 PM</td>
<td>Moscone South – 103</td>
</tr>
<tr>
<td>CON4666</td>
<td>Oracle Enterprise Manager 12c Database Lifecycle Management Automatic Provisioning and Patching</td>
<td>Wednesday</td>
<td>3:30 PM</td>
<td>Moscone South – 131</td>
</tr>
<tr>
<td>CON8768</td>
<td>DBA Best Practices for Protecting Data Privacy with Oracle’s Data Masking</td>
<td>Wednesday</td>
<td>3:30 PM</td>
<td>Moscone West – 2024</td>
</tr>
<tr>
<td>CON9577</td>
<td>Active Session History Deep Dive: Advanced Performance Analysis Tips</td>
<td>Wednesday</td>
<td>3:30 PM</td>
<td>Moscone South – 104</td>
</tr>
<tr>
<td>CON3255</td>
<td>Being Sure: Confident Consolidations with Oracle Real Application Testing 12c</td>
<td>Wednesday</td>
<td>5:00 PM</td>
<td>Moscone South – 306</td>
</tr>
<tr>
<td>CON3103</td>
<td>Real Oracle Real Application Testing: What to Expect and Prepare For</td>
<td>Thursday</td>
<td>12:30 PM</td>
<td>Moscone South - 104</td>
</tr>
</tbody>
</table>