What to expect from the Optimizer when upgrading from Oracle Database 10g to 11g

Maria Colgan & Mohamed Zait
Agenda

• Changes in behaviour
• SQL Plan Management
• Pre-upgrade checklist
• Post-upgrade checklist
• Correcting regressed SQL Statements
Changes in behavior
## Init.ora Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>In 10g</th>
<th>In 11g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizer_mode</td>
<td>Cost-based Optimizer used for all SQL Statements</td>
<td>All_rows</td>
<td>All_rows</td>
</tr>
<tr>
<td>Optimizer_Dynamic_Sampling</td>
<td>If no statistics on an object automatically gathered at parse</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Optimizer_Secure_view_merging</td>
<td>Additional security checks before merging a view</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>Optimizer_use_invisible_indexes</td>
<td>Allows Optimizer to use an invisible index as access method</td>
<td>N/A</td>
<td>False</td>
</tr>
<tr>
<td>Optimizer_use_pending_statistics</td>
<td>Allows Optimizer to use an pending statistics</td>
<td>N/A</td>
<td>False</td>
</tr>
<tr>
<td>Optimizer_capture_SQL_plan_baselines</td>
<td>Automatically captures execution plans into SPM</td>
<td>N/A</td>
<td>False</td>
</tr>
<tr>
<td>Optimizer_use_SQL_plan_baselines</td>
<td>Optimizer uses any existing SQL Plan Baseline</td>
<td>N/A</td>
<td>True</td>
</tr>
</tbody>
</table>
## New DBMS_STATS Subprograms

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Function</th>
<th>In 10gR2</th>
<th>In 11g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather_System_Stats</td>
<td>Gathers stats on CPU and IO speed of H/W</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Gather_Dictionary_Stats</td>
<td>Gathers stats on dictionary objects</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Gather_Fixed_Object_Stats</td>
<td>Gather stats on V$views</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Publish_Pending_stats</td>
<td>Pending stats allows stats to be gather but not published immediate</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Restore_Table_Stats</td>
<td>Revert stats back to what they were before</td>
<td>10.2.0.4</td>
<td>Yes</td>
</tr>
<tr>
<td>Diff_Table_Stats</td>
<td>Compare stats for a table from two different sources</td>
<td>10.2.0.4</td>
<td>Yes</td>
</tr>
<tr>
<td>Create_Extended_stats</td>
<td>Gathers stats for a user specified column group or an expression</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>Set_Table_Perfs</td>
<td>Sets stats preferences of a table</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Automatic Statistics Gathering Job

- Introduced in 10g
- Gathers statistics on objects where
  - Statistics are missing
  - Statistics are stale
- In 10g its an Oracle Scheduler job
  - Runs during maintenance window
- In 11g its an Autotask
  - Runs during maintenance window
  - Use `DBMS_AUTO_TASK_ADMIN` package to control job
New Features

• New Optimizations
  – Group-by placement
  – Enhanced join predicate push down
  – Null-aware antijoin

• Adaptive Cursor Sharing (enhanced bind peeking)

• Extended Statistics
  – Multi-column statistics (for correlation)
  – Statistics on expressions

• Pending Statistics

Each new feature could potentially change a plan

How can you maintain performance -> stability during upgrade?

SQL Plan Management
SQL Plan Management

Prior to 11g
- Unpredictable changes can happen to an execution plan
- Avoiding plan changes the only method to avoid performance regression
  - Lock Statistics to prevent them from changing
  - Freezing an execution plan with a Stored Outline
- No mechanism for plans to evolve

Solution
- Optimizer automatically manages ‘execution plans’
  - Only known and verified plans are used
- Plan changes are verified
  - Only comparable or better plans are used going forward

SQL Plan Management is controlled plan performance
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

Invoke or schedule verification

Optimizer checks if new plan is as good as or better than old plan

Plans which perform as good as or better than original plan are added to the plan baseline

Plans which don’t perform as good as the original plan stay in the plan history and are marked unaccepted
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
Pre-Upgrade Steps
Pre-Upgrade Step

• 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 SPA

Before change

O_F_E=10

Oracle Database 11g

Regressing statements

Plan History

Baseline

GB
HJ

GB
HJ

No plan regressions

After change

O_F_E=11

optimizer_features_enable

SQL Performance Analyzer

Oracle Database 10g

Well tuned plans
Capturing Plans using Stored outlines

1. Begin with 
   `CREATE_STORED_OUTLINES=true`

2. Run all SQL in the Application and auto create a Stored Outline for each one

3. After Store Outlines are captured 
   `CREATE_STORED_OUTLINES=false`

4. Upgrade to 11g

5. Migrate Stored Outlines into SPM
Capturing Plans using SQL Tuning Set

1. Create STS for critical statements
2. Upgrade to 11g
3. Bulk load plans into SPM

Database Upgrade

Oracle Database 11g

No plan regressions

Plan History

Plan

Baseline

Well tuned plan

Oracle Database 11g

Oracle Database 10g

DBA
Capturing Plans Using an 11g test environment

1. Create baselines from tuned stmts
2. Create staging table & pack baselines into it
3. Export staging table
4. Import staging table
5. Unpack baselines into SPM

Production Database 11g

Development / Test Database 11g
Post-Upgrade Steps
Post-upgrade Steps

- Load SPM with 10g plans
  - From a STS create in Oracle Database 10gR2
  - From Stored Outlines
  - From SQL Tuning Set
  - From a staging table
  - From the Cursor Cache

- Manage Optimizer Statistics
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

- 2. Run all SQL in the Application and auto load SQL Plan Baselines with 10g plan
- 3. Auto Capture 10g plans
- 4. After plans are loaded change OFE to 11
- 5. 11g plan queue for verification
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();`
Correcting Regressed SQL Statements
Correcting Regressed SQL Statement

- Load plans from a SQL Tuning Set
- Load plans from the Cursor Cache
- Load plans from Stored Outlines
- Load plans from a staging table
Upgrade Demo
Correcting Regressed SQL Statement

- Load plans from a SQL Tuning Set
- Load plans from the Cursor Cache
- Load plans from Stored Outlines
- Load plans from a staging table
- Load a hinted execution plan
For More Information

search.oracle.com

Upgrading Optimizer

or

http://www.oracle.com/technology/products/bi/db/11g/pdf/twp_upgrading_10g_to_11g_what_to_expect_from_optimizer.pdf
The preceding 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.