Adaptive Optimization
Presented by: Kerry Osborne
Hotsos Symposium 2013
whoami -

Never Worked for Oracle
Worked with Oracle DB Since 1982 (V2)
Working with Exadata since early 2010
Work for Enkitec (www.enkitec.com)
(Enkitec owns several Exadatan – V2/X2/X3)
(And Others BDA, Exalytics, ODA, etc…)
Worked on a couple of books
Hadoop Aficionado
Exadata Fan Boy
This is my 11th Presentation at Hotsos 😊

Blog: kerryosborne.oracle-guy.com
Twitter: @KerryOracleGuy
Top Secret Feature of BDA
What I Did Last Week
What’s the Point?

Sometimes the Optimizer Makes Mistakes
It’s Often Pretty Easy to Spot the Mistakes
Why Not Let the DB Fix the Mistakes on the Fly?
How Does the Optimizer Mess Up?

Cardinality – Misunderestimate

mostly …
and it’s pretty easy to recognize …

Estimated Rows ≠ Actual Rows
Cardinality - Misunderestimate

```
PLAN_TABLE_OUTPUT

SQL_ID 0qa98gcnnza7h, child number 1

select avg(pk_col) from kso.skew where col1 > 0

Plan hash value: 568322376

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Starts</th>
<th>E-Rows</th>
<th>A-Rows</th>
<th>A-Time</th>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1:00:00:06.43</td>
<td>162K</td>
</tr>
<tr>
<td>1</td>
<td>SORT AGGREGATE</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1:00:00:06.43</td>
<td>162K</td>
</tr>
<tr>
<td>*2</td>
<td>TABLE ACCESS STORAGE FULL</td>
<td>SKEW</td>
<td></td>
<td>1234</td>
<td>32M</td>
<td>0:00:00:03.43</td>
<td>162K</td>
</tr>
</tbody>
</table>
```
Cardinality - Misunderestimate

• Wolfgang Breitling – Tuning By Cardinality Feedback
• Randolf Geist – xplan_extended_display_cursor.sql
  • Adrian Billington – Xplan Wrapper
  • Kyle Halley – Display_Cursor Post
Optimizer Evolution

- Bind Variable Peeking
- Dynamic Sampling
- Adaptive Cursor Sharing
- Cardinality Feedback
- Tuning Advisor

- Trend towards more dynamic plans
- 11g drawbacks
  - must run badly before it does anything
  - “fixes” – not persisted
Oracle 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. Release timing for Oracle Database 12c is planned for Calendar Year 2013.
Adaptive Optimization

Adaptive Query Optimization

Adaptive Plans
- Join Methods
- Parallel Distribution Methods

Adaptive Statistics
- At Compile Time
- At Run Time

Cardinality Feedback
Dynamic Sampling
But First - Some New Terms

• Adaptive Optimization – any dynamic change to plan

• Adaptive Plans – changed from default on 1st execution
• Automatic Re-optimization – 2nd execution
• Statistics Feedback = Cardinality Feedback
• Dynamic Statistics = Dynamic Sampling
• SQL Plan Directives = Persisted Dynamic Sampling (for now)
  SPD = PDS

Note
-----
- dynamic statistics used: dynamic sampling (level=2)
- statistics feedback used for this statement
- this is an adaptive plan
- 2 Sql Plan Directives used for this statement
Adaptive Execution Plans
Join Methods

Adaptive Query Optimization

Adaptive Plans
- Join Methods
- Parallel Distribution Methods

Adaptive Statistics
- At Compile Time
- At Run Time
Adaptive Execution Plans

Join Methods

- Optimizer Can Change It’s Mind in Mid-Execution

- 2 Join Methods
  - Nested Loop
  - Hash Join
Adaptive Optimization

Controls

optimizer_adaptive_features = false
- big switch - controls all adaptive stuff

optimizer_features_enable <= 12.1.0.1
- even bigger switch – please don’t use this one!

optimizer_adaptive_reporting_only = true

_optimizer_adaptive_plans=false
- individual control for adaptive plans

_optimizer_use_feedback=false
- individual control for cardinality feedback

optimizer_dynamic_sampling=0
- individual control for dynamic sampling
Adaptive Execution Plans

Alternative sub-plans are pre-computed
Sub-plans stored in the cursor
Stats collector inserted before join
Rows buffered until final decision is made

Rows coming out via inner nested loop are buffered up to a point. If row count exceeds threshold then switch to hash join.
Adaptive Execution Plans

- Number of rows seen in statistics collector exceeds threshold
- Plan switches to hash join
- Statistics collector disabled
- Plan resolved on first execution & remains the same for subsequent executions

Final Plan is a hash join
Adaptive Execution Plans
Finding Them (is easy)

SYS@BETA3> select sql_id, child_number, sql_text from v$sql
 2   where IS_REOPTIMIZABLE = 'Y'
 3   and IS_RESOLVED_ADAPTIVE_PLAN = 'Y'
 4   order by 1;

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>SQL_TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0643yhacr145x</td>
<td></td>
<td>0 SELECT OPR.NAME, MAX(NVL(DBC LOADS,0)) LOADS, MAX(NVL(FU FEATURE USED,1)) USED FROM SYS.GV$DB_OBJECT_CACHE DBC,CTXSYS.DR$FEATURE_USED FU, ( SELECT UO.OBJECT_NAME NAME FROM ALL_OBJECTS UO,CTXSYS.DR$DBO DBO WHERE UO.OWNER = 'CTXSYS' AND DBO.OWNER = OBJECT_NAME AND DBO.OPERATOR = 'OPERATOR' ) OPR WHERE OPR.OPERATOR = FU.FEATURE_NAME(+) AND OPR.OPERATOR = DBO.NAME(+) AND FU.FEATURE_E_TYPE(+) = 2 GROUP BY OPR.OPERATOR ORDER BY OPR.OPERATOR ASC</td>
</tr>
<tr>
<td>0ghr54snhw89c</td>
<td></td>
<td>0 SELECT COUNT(*) FROM DBA_OBJ_AUDIT_OPTS</td>
</tr>
<tr>
<td>0v37jgm4mdnjw</td>
<td></td>
<td>0 select count(*) from dba_sequences where sequence_owner != 'SYS' and session_flag = 'N'</td>
</tr>
</tbody>
</table>
Digression - OTHER_XML

SYS@BETA3> @other_xml
Enter value for sql_id: fq5171y68rx1q
Enter value for child_number: 0

OTHER_XML

<other_xml><info type="adaptive_plan">yes</info><info type="db_version">12.1.0.1</info><info type="parse_schema"><![CDATA["SYS"]]>\</info><info type="dynamic_sampling">2</info><info type="plan_hash">1015358205</info><info type="plan_hash_2">3087610831</info><spc><cv>8</cv><cu>2</cu></spc><display_map><row op="1" dis="1"...
Adaptive Execution Plans
Displaying Default & Final Plans

Default – EXPLAIN PLAN + DBMS_XPLAN.DISPLAY

Default – Turn Off Feature - Standard DBMS_XPLAN.DISPLAY_CURSOR

Final - Standard DBMS_XPLAN.DISPLAY_CURSOR

Mixed - Use DBMS_XPLAN – with format “adaptive +report”

```sql
select * from table(dbms_xplan.display_cursor('&sql_id','&child_no','adaptive +report'));
```

Produces plan which shows steps which were abandoned in final plan. Abandoned steps are marked with a “-”
Adaptive Execution Plans
Displaying Default Plan

```
SYSEBETA3> alter session set optimizer_adaptive_features=false;
Session altered.
SYSEBETA3> -- run query here
SYSEBETA3> select * from table(dbs_xplan.display_cursor(null,null,'RUNSTATS_LAST'));
```

PLAN_TABLE_OUTPUT

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>654utuvy6fz5w, child number 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>select product_name from oe.order_items o, oe.product_information p</td>
<td></td>
</tr>
<tr>
<td>where o.unit_price=15 and o.quantity &gt; 1 and p.product_id = o.product_id</td>
<td></td>
</tr>
</tbody>
</table>

Plan hash value: 1255158658

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 3</td>
<td>TABLE ACCESS FULL</td>
<td>ORDER_ITEMS</td>
<td>4</td>
<td>128</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 4</td>
<td>INDEX UNIQUE SCAN</td>
<td>PRODUCT_INFORMATION_PK</td>
<td>1</td>
<td>48</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PRODUCT_INFORMATION</td>
<td>1</td>
<td>20</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

3 - filter("O"."UNIT_PRICE"=15 AND "O"."QUANTITY">1))
4 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")

24 rows selected.
Adaptive Execution Plans
Displaying Final Plan

SYS@BETA3> select * from table(dbms_xplan.display_cursor('654utuvy6fz5w',0));

PLAN_TABLE_OUTPUT
-----------------------------------------------------------------
SQL_ID  654utuvy6fz5w, child number 0
-----------------------------------------------------------------
select product_name from oe.order_items o, oe.product_information p
where o.unit_price=15 and o.quantity > 1 and p.product_id = o.product_id

Plan hash value: 1553478007

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>8</td>
<td></td>
<td>8 (100)</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>HASH JOIN</td>
<td></td>
<td>13</td>
<td>416</td>
<td>8 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>TABLE ACCESS</td>
<td>ORDER_ITEMS</td>
<td>13</td>
<td>156</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS</td>
<td>PRODUCT_INFORMATION</td>
<td>288</td>
<td>5760</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):
------------------------------------------------------

 1 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
 2 - filter("O"."UNIT_PRICE"=15 AND "O"."QUANTITY">1))

Note
----
- this is an adaptive plan
Adaptive Execution Plans
Displaying Adaptive Plans

SYS@BETA3> select * from table(dbms_xplan.display_cursor('654utuvv6fz5w', 0, 'adaptive +report'));

PLAN_TABLE_OUTPUT
SQL_ID 654utuvv6fz5w, child number 0

select product_name from oe.order_items o, oe.product_information p
where o.unit_price=15 and o.quantity > 1 and p.product_id = o.product_id

Plan hash value: 1553478007

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
<td>13</td>
<td>416</td>
<td>8 (100)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>NESTED LOOPS</td>
<td></td>
<td>13</td>
<td>416</td>
<td>8 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>NESTED LOOPS</td>
<td></td>
<td>13</td>
<td>416</td>
<td>8 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TABLE ACCESS FULL</td>
<td>ORDER_ITEMS</td>
<td>13</td>
<td>156</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>INDEX UNIQUE SCAN</td>
<td>PRODUCT_INFORMATION_PK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PRODUCT_INFORMATION</td>
<td>1</td>
<td>20</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>8</td>
<td>TABLE ACCESS FULL</td>
<td>PRODUCT_INFORMATION</td>
<td>288</td>
<td>5760</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

<table>
<thead>
<tr>
<th>Id</th>
<th>Predicate Type</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>access</td>
<td>&quot;PRODUCT_ID&quot;=&quot;0&quot;.&quot;PRODUCT_ID&quot;</td>
</tr>
<tr>
<td>5</td>
<td>filter</td>
<td>&quot;UNIT_PRICE&quot;=15 AND &quot;QUANTITY&quot;&gt;1</td>
</tr>
<tr>
<td>6</td>
<td>access</td>
<td>&quot;PRODUCT_ID&quot;=&quot;0&quot;.&quot;PRODUCT_ID&quot;</td>
</tr>
</tbody>
</table>

Note:
- this is an adaptive plan (rows marked '-' are inactive)
Adaptive Execution Plans
Displaying Adaptive Plans (+report)

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH JOIN</td>
<td></td>
<td>4</td>
<td>128</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>TABLE ACCESS FULL</td>
<td>ORDER_ITEMS</td>
<td>4</td>
<td>48</td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>PRODUCT_INFORMATION</td>
<td>1</td>
<td>20</td>
<td>1 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
2 - filter("O"."UNIT_PRICE"=15 AND "O"."QUANTITY">1))

Note
- this is an adaptive plan
Adaptive Execution Plans
Displaying Adaptive Plans (+report)

Reoptimized plan:

This cursor is marked for automatic reoptimization, but automatic reoptimization is enabled for reporting mode only. The plan that would be selected on the next execution if automatic reoptimization were enabled is displayed below.

Plan hash value: 1353478007

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>32</td>
<td>8</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 1</td>
<td>HASH JOIN</td>
<td></td>
<td>1</td>
<td>32</td>
<td>8</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>TABLE ACCESS FULL</td>
<td>ORDER_ITEMS</td>
<td>13</td>
<td>156</td>
<td>3</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>PRODUCT_INFORMATION</td>
<td>288</td>
<td>5760</td>
<td>5</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - access("P"."PRODUCT_ID"="0"."PRODUCT_ID")
2 - filter("0"."UNIT_PRICE"=15 AND "0"."QUANTITY">1)

Note
---
- this is an adaptive plan
Adaptive Execution Plans

SPM Interaction

Baselines Behave Pretty Much As You’d Expect

Adaptive Plans Can Be Captured (The Final Plan)

Once SQL Using Baseline – No Longer Marked Adaptive

If Capture Is On – Unaccepted Plans Flagged as Adaptive

* Note: Do Baselines Actually Store Plans Now – Not Just Hints ?
  • Maria Says Yes!
  • Seems to be true!
Adaptive Execution Plans

SPM Interaction

---

**PLAN_TABLE_OUTPUT**

**SQL_ID** 654utuvy6fz5w, child number 1

```
select product_name from oe.order_items o, oe.product_information p
where o.unit_price=15 and o.quantity > 1 and p.product_id = o.product_id
```

Plan hash value: 1553478007

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>8</td>
<td></td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>* 1</td>
<td>HASH JOIN</td>
<td></td>
<td>13</td>
<td>416</td>
<td>0</td>
<td>00:00:01</td>
</tr>
<tr>
<td>* 2</td>
<td>TABLE ACCESS FULL</td>
<td>ORDER_ITEMS</td>
<td>13</td>
<td>156</td>
<td>0</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>PRODUCT_INFORMATION</td>
<td>288</td>
<td>5760</td>
<td>0</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - access("P"."PRODUCT_ID"="O"."PRODUCT_ID")
2 - filter("O"."UNIT_PRICE"=15 AND "O"."QUANTITY">1))

Note
---

- SQL plan baseline SQLID_654utuvy6fz5w_1553478007 used for this statement
Adaptive Execution Plans
Parallel Distribution Methods

Adaptive Query Optimization

Adaptive Plans
- Join Methods
- Parallel Distribution Methods

Adaptive Statistics
- At Compile Time
- At Run Time
Adaptive Distribution Methods

- New adaptive distribution method HYBRID-HASH
  - Statistic collectors inserted in front of PX process
  - If actual number of rows less than threshold, switch from HASH to Broadcast
    - Threshold number of total rows < 2x DOP

- Enabled by default
Adaptive Distribution Methods

- Distribution method decision based on expected number of rows
- Cardinality based distribution skew is common
- Can result in very uneven distribution

Diagram:
- Data source connected to distribution methods
- Methods include p1, p2, p3, p4, p5, p6, p8
- Statistics node
- Hybrid-Hash and Broadcast connections

Hybrid-Hash
Broadcast
Adaptive Statistics
Dynamic Statistics (Sampling)

Adaptive Query Optimization

Adaptive Plans
- Join Methods
- Parallel Distribution Methods

Adaptive Statistics
- At Compile Time
- At Run Time

Dynamic Sampling
Dynamic Statistics

- Dynamic statistics are used to compensate for missing, stale, or incomplete statistics.
- They can be used for table scans, index access, and joins.
- Optimizer computes a time budget for generating dynamic statistics based on query run-time.
- Statistics are stored in memory and can be shared across queries.
- My Blog: Randolf Geist on Dynamic Sampling.
Adaptive Statistics
Dynamic Statistics (Sampling)

Adaptive Query Optimization

Adaptive Plans
- Join Methods
- Parallel Distribution Methods

Adaptive Statistics
- At Compile Time
- At Run Time

Cardinality Feedback
Adaptive Statistics
Re-optimization

- During execution optimizer estimates are compared to execution statistics
- If statistics vary significantly then a new plan will be chosen for subsequent executions based on execution statistics
- Re-optimization uses statistics gathered from previous executions
- First introduced as Cardinality Feedback in 11.2
Adaptive Statistics
Cardinality Feedback - 11g

- Statistics gathered about data volume and data type seen during execution
- If execution statistics vary significantly statement will be hard parsed on the next execution using the execution statistics instead
- Statements are only monitored once if they don’t show significant differences initially they won’t change in the future
- Only individual table cardinalities and group by estimates examined - not joins
- Information is stored in the cursor only and is lost if cursor ages out
Adaptive Statistics
New Re-optimization

- Join statistics are also monitored
- Works with adaptive cursor sharing for statement with binds
- New Column in V$SQL IS_REOPTIMIZABLE
- Information found at execution time is persisted as SQL Plan Directives
“SPD are objects generated automatically by Oracle. For example, if Oracle detects that the single table cardinality estimated made by the optimizer is different from the actual number of rows returned when accessing the table, it will automatically create a directive to perform dynamic statistics for the table. When any SQL statement referencing the table is compiled, the optimizer will perform dynamic statistics for the table to get a more accurate estimate.”

~ PL/SQL Packages Reference (12c Release 1)

SPD = PDS (Persisted Dynamic Sampling)
## SQL Plan Directives

The table below shows the columns and their comments for the directive table in the database.

### Column Name and Comments

<table>
<thead>
<tr>
<th>COLUMN_NAME</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIRECTIVE_ID</td>
<td>The identifier of the sql plan directive</td>
</tr>
<tr>
<td>TYPE</td>
<td>The type of the sql plan directive</td>
</tr>
<tr>
<td>STATE</td>
<td>The state of the sql plan directive</td>
</tr>
<tr>
<td>AUTO_DROP</td>
<td>If YES, the sql plan directive gets dropped when unused beyond SPD_RETENTION_WEEKS</td>
</tr>
<tr>
<td>REASON</td>
<td>The reason for creating the sql plan directive</td>
</tr>
<tr>
<td>CREATED</td>
<td>The creation timestamp of the sql plan directive</td>
</tr>
<tr>
<td>LAST_MODIFIED</td>
<td>The timestamp of most recent modification of the sql plan directive</td>
</tr>
<tr>
<td>LAST_USED</td>
<td>The timestamp of most recent usage of the sql plan directive</td>
</tr>
</tbody>
</table>

---

8 rows selected.

### Additional Columns and Comments

<table>
<thead>
<tr>
<th>COLUMN_NAME</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWNER</td>
<td>The username of the owner of the object in the sql plan directive</td>
</tr>
<tr>
<td>OBJECT_NAME</td>
<td>The name of the object in the sql plan directive</td>
</tr>
<tr>
<td>SUBOBJECT_NAME</td>
<td>The name of the sub-object (for example column) in the sql plan directive</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>The type of the (sub-)object in the sql plan directive</td>
</tr>
<tr>
<td>NOTES</td>
<td>Other notes about the object</td>
</tr>
</tbody>
</table>
### SQL Plan Directives

```sql
SYS@BETA3> @directive_objs
Enter value for object_name: ORDER_ITEMS

<table>
<thead>
<tr>
<th>DIRECTIVE_ID</th>
<th>OWNER</th>
<th>OBJECT_NAME</th>
<th>SUBOBJECT_NAME</th>
<th>OBJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>14460712757220495343</td>
<td>OE</td>
<td>ORDER_ITEMS</td>
<td>UNIT_PRICE</td>
<td>COLUMN</td>
</tr>
<tr>
<td>OE</td>
<td>ORDER_ITEMS</td>
<td>QUANTITY</td>
<td>COLUMN</td>
<td>COLUMN</td>
</tr>
</tbody>
</table>

SYS@BETA3> select directive_id, type, state, reason, created
2 from dba_sql_plan_directives
3 where directive_id like nvl('&directive_id', directive_id);
Enter value for directive_id: 14460712757220495343

<table>
<thead>
<tr>
<th>DIRECTIVE_ID</th>
<th>TYPE</th>
<th>STATE</th>
<th>REASON</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>14460712757220495343</td>
<td>DYNAMIC_SAMPLING</td>
<td>HAS_STATS</td>
<td>SINGLE TABLE CARDINALITY MIESTIMATE 04-MAR-13 11.15.38.000000 PM</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DIRECTIVE_ID</th>
<th>OWNER</th>
<th>OBJECT_NAME</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>14460712757220495343</td>
<td>OE</td>
<td>ORDER_ITEMS</td>
<td>&lt;obj_note&gt;</td>
</tr>
</tbody>
</table>
```

*<equality_predicates_only:NO</equality_predicates_only>*
*<simple_column_predicates_only:YES</simple_column_predicates_only>*
*<index_access_by_join_predicates:NO</index_access_by_join_predicates>*
*<filter_on_joining_object:NO</filter_on_joining_object>*
*<obj_note>*/
SQL Plan Directives

```
SYS@BETA3> select directive_id, type, state, reason, created 
  2  from dba_sql_plan_directives 
  3  where directive_id like nvl('&directive_id', directive_id);
```

Enter value for directive_id: 14460712757220495343

<table>
<thead>
<tr>
<th>DIRECTIVE_ID</th>
<th>TYPE</th>
<th>STATE</th>
<th>REASON</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>14460712757220495343</td>
<td>DYNAMIC_SAMPLING HAS_STATS</td>
<td>HAS_STATS</td>
<td>SINGLE TABLE CARDINALITY MISESTIMATE</td>
<td>04-MAR-13 11.15.38.000000 PM</td>
</tr>
</tbody>
</table>
SQL Plan Directives

```sql
SYS@BETA3> select distinct type, reason, state from DBA_SQL_PLAN_DIRECTIVES order by 1,2;

<table>
<thead>
<tr>
<th>TYPE</th>
<th>REASON</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DYNAMIC_SAMPLING GROUP BY CARDINALITY MISESTIMATE</td>
<td>HAS_STATS NEW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>JOIN CARDINALITY MISESTIMATE</td>
<td>HAS_STATS NEW PERMANENT</td>
</tr>
<tr>
<td></td>
<td>SINGLE TABLE CARDINALITY MISESTIMATE</td>
<td>HAS_STATS MISSING_STATS NEW PERMANENT</td>
</tr>
</tbody>
</table>

SYS@BETA3> select state, count(*) from DBA_SQL_PLAN_DIRECTIVES group by state;

<table>
<thead>
<tr>
<th>STATE</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERMANENT</td>
<td>38</td>
</tr>
<tr>
<td>MISSING_STATS</td>
<td>7</td>
</tr>
<tr>
<td>HAS_STATS</td>
<td>68</td>
</tr>
<tr>
<td>NEW</td>
<td>49</td>
</tr>
</tbody>
</table>
```
SQL Plan Directives

States

NEW - 1st pass

MISSING_STATS - needs extended stats
  (gathered automagically)

HAS_STATS – extended stats have now been gathered
  (Intermediate State – new statements may still need SPD’s)

PERMANENT - extended stats have now been gathered
  (but SPD still needed because of != predicates)
SQL Plan Directives

Management

Managed with DBMS_SPD
- but not really much to manage
- can put them in a staging table and move them to another DB
- can flush any in memory to disk (flushed every 15m by default)
- can drop specific directives
Well, How Did We Get Here?

The New Optimizer
Wrap Up

Even More Automagical Stuff

Name Changes Can be Confusing
- “statistics” happy in naming
- Dynamic Sampling = Dynamic Statistics
- Statistics Feedback = Cardinality Feedback

Ideas are Sound
- learn from execution statistics
- eliminate “must run bad first” behavior
- add persistence

It’s the Default – so you will see it
Questions?

Contact Information  : Kerry Osborne

kerry.osborne@enkitec.com
kerryosborne.oracle-guy.com
www.enkitec.com