Adaptive Optimization

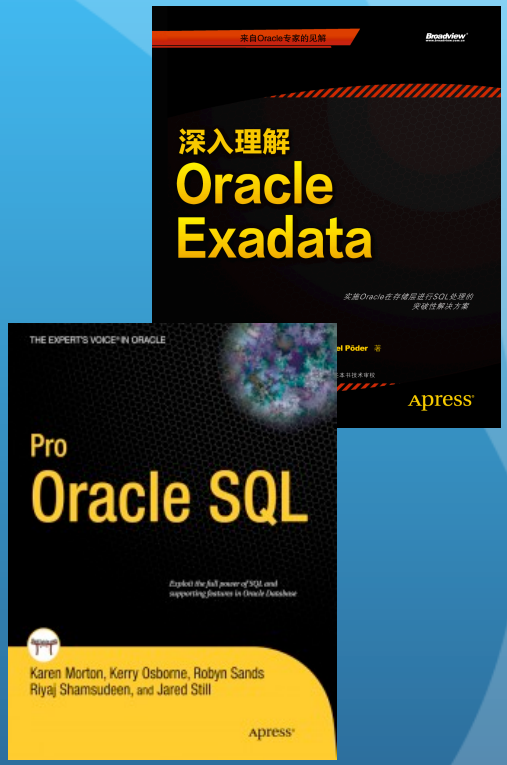
Presented by: Kerry Osborne

Hotsos Symposium 2013

enkitec

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

enkitec

Top Secret Feature of BDA

enkitec

What I Did Last Week

enkitec

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?

enkitec

How Does the Optimizer Mess Up?

Cardinality – Misunderestimate

mostly …
and it's pretty easy to recognize …

Estimated Rows ≠ Actual Rows

enkitec

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

-------------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |      |      1 |        |      1 |00:00:06.43 |     162K|
|   1 |  SORT AGGREGATE             |      |      1 |      1 |      1 |00:00:06.43 |     162K|
|*  2 |   TABLE ACCESS STORAGE FULL| SKEW |      1 |   1234 |    32M|00:00:03.43 |     162K|
-------------------------------------------------------------------------------------------------------
```

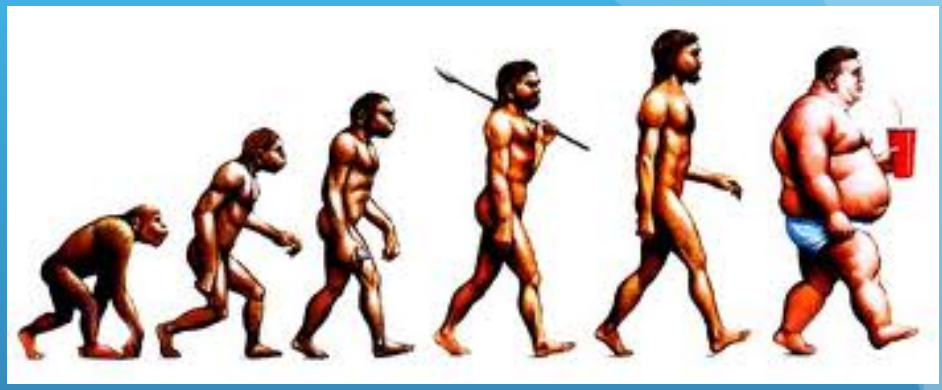
Cardinality – Misunderestimate

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

enkitec

Optimizer Evolution

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



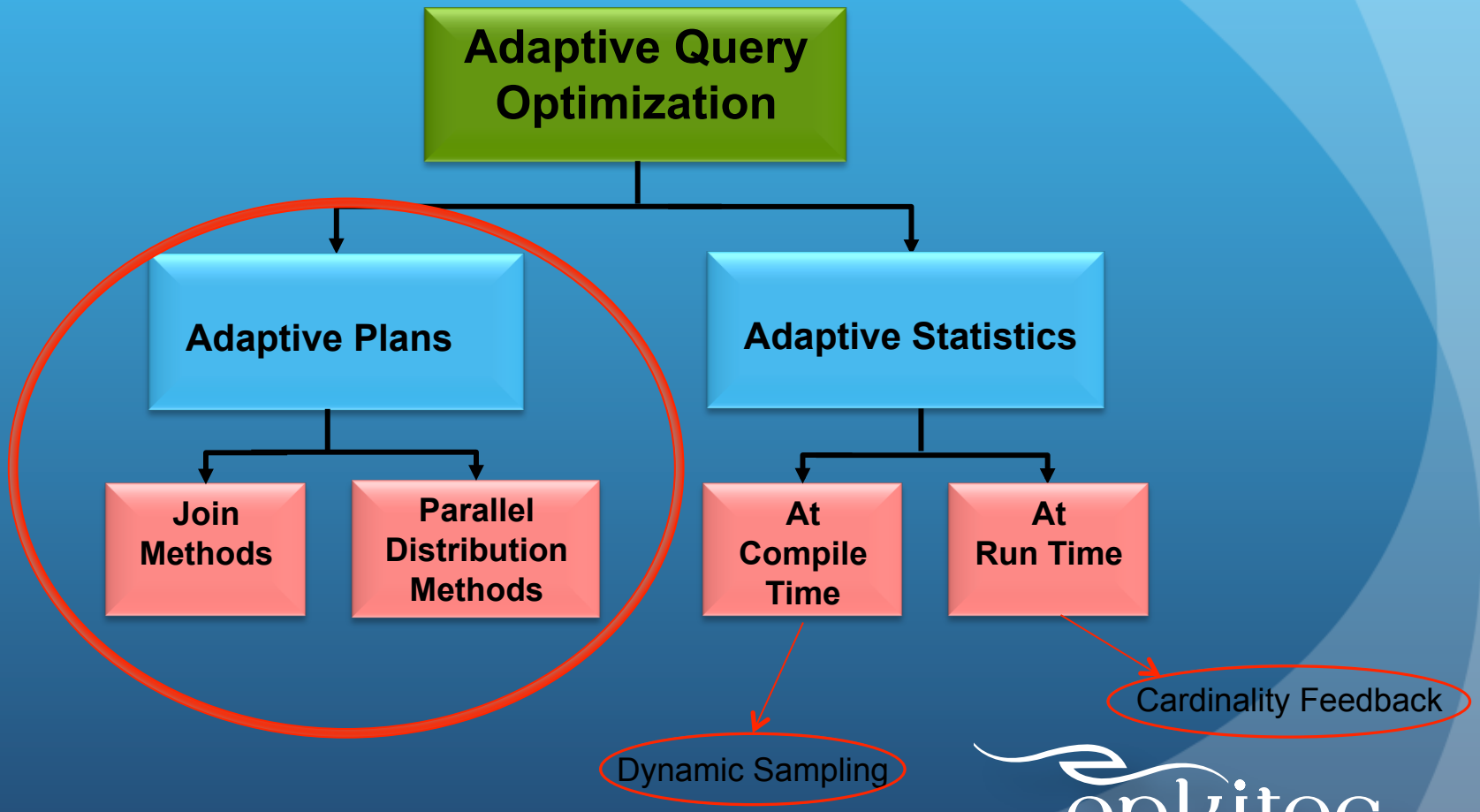
- Trend towards more dynamic plans
- 11g drawbacks
  - must run badly before it does anything
  - “fixes” – not persisted

enkitec

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



enkitec

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)

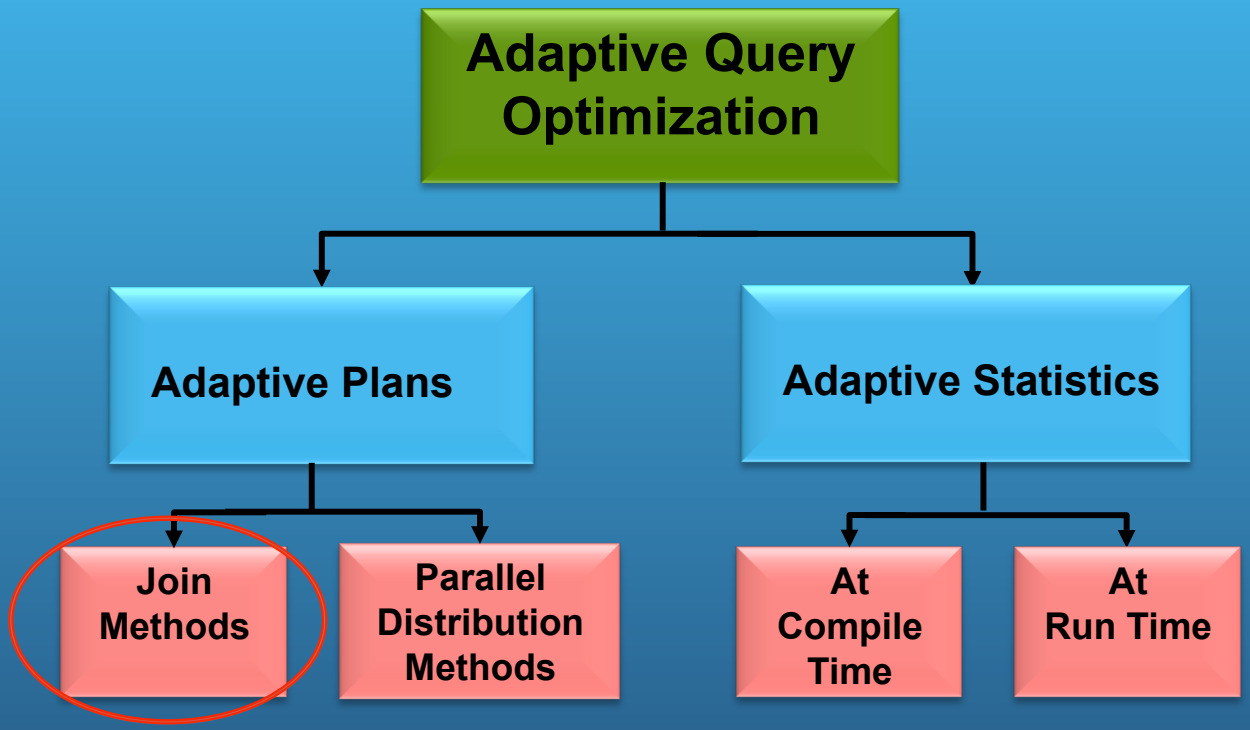
SPD = PDS

```
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
```

enkitec

Adaptive Execution Plans

Join Methods



enkitec

Adaptive Execution Plans

Join Methods

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

- 2 Join Methods
  - Nested Loop
  - Hash Join

enkitec

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

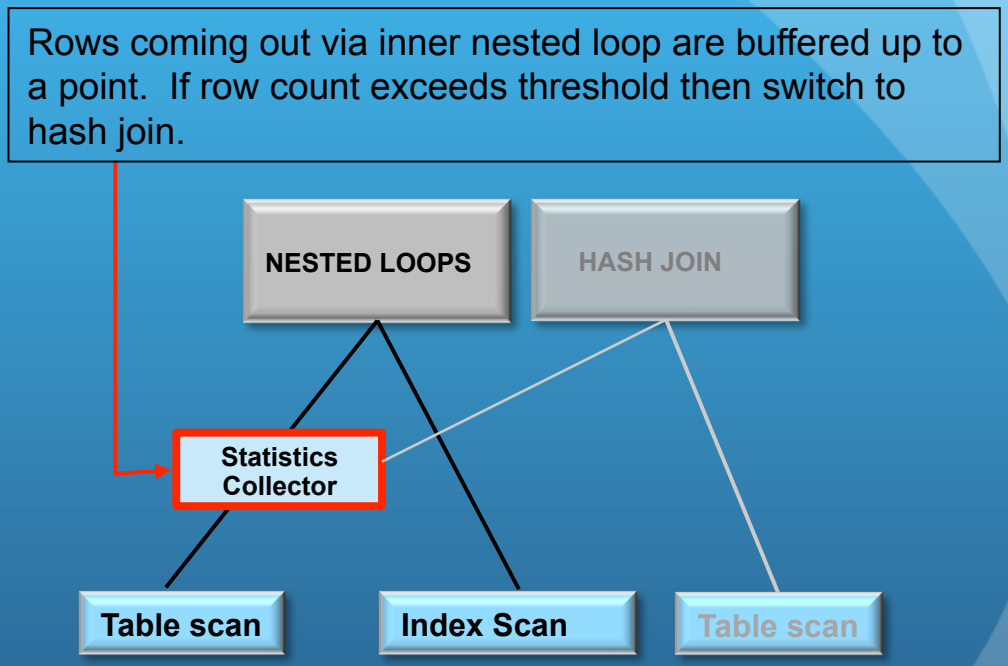
_optimizer_adaptive_plans=false
- individual control for adaptive plans

_optimizer_use_feedback=false
- individual control for cardinality feedback

optimizer_dynamic_sampling=0
- individual control for dynamic sampling

Adaptive Execution Plans

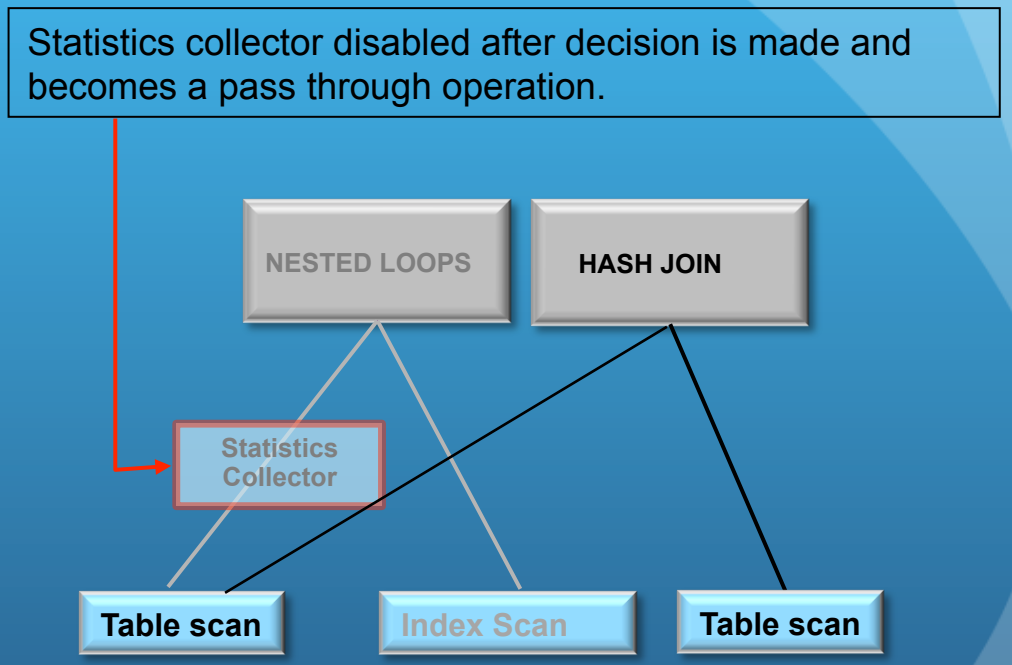
- Alternative sub-plans are pre-computed
- Sub-plans stored in the cursor
- Stats collector inserted before join
- Rows buffered until final decision is made



enkitec

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

enkitec

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;

SQL_ID        CHILD_NUMBER SQL_TEXT
------------- ------------ --------------------------------------------------------------------------------
0643yhacr145x            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_NAME = OBJECT_NAME AND DBO_TYPE = 'OPERATOR' AND OBJECT_TYPE = 'OPERATOR' )
                           OPR WHERE OPR.NAME = FU.FEATURE_NAME(+) AND OPR.NAME = DBC.NAME(+) AND FU.FEATUR
                           E_TYPE(+) = 2 GROUP BY OPR.NAME ORDER BY OPR.NAME ASC

0ghr54snhw89c            0 SELECT COUNT(*) FROM DBA_OBJ_AUDIT_OPTS
0v37jgm4mdnjw            0 select count(*) from dba_sequences where sequence_owner != 'SYS' and session_fla
                           g = 'N'
```

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_sam
pling">2</info><info type="plan_hash">1015358205</info><info type="plan_hash_2">
3087610831</info><spd><cv>8</cv><cu>2</cu></spd><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”

```
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

```
SYS@BETA3> alter session set optimizer_adaptive_features=false;

Session altered.

SYS@BETA3> --> run query here

SYS@BETA3> select * from table(dbms_xplan.display_cursor(null,null,'RUNSTATS_LAST'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  654utuvy6fz5w, child number 3
-------------------------------------
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: 1255158658

-----------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                            |       |       |     7 (100)|          |
|   1 |  NESTED LOOPS                |                            |       |       |            |          |
|   2 |   NESTED LOOPS               |                            |     4 |   128 |     7   (0)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL         | ORDER_ITEMS                |     4 |    48 |     3   (0)| 00:00:01 |
|*  4 |    INDEX UNIQUE SCAN         | PRODUCT_INFORMATION_PK     |     1 |       |     0   (0)|          |
|   5 |   TABLE ACCESS BY INDEX ROWID| PRODUCT_INFORMATION        |     1 |    20 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------------------

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

-------------------------------------------------------------------------------------------
| Id  | Operation          | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |                     |       |       |     8 (100)|          |
|*  1 |  HASH JOIN         |                     |    13 |   416 |     8   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| ORDER_ITEMS         |    13 |   156 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| PRODUCT_INFORMATION |   288 |  5760 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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('654utuvy6fz5w',0,'adaptive +report'));

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

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

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

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

Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```

Abandoned

Adaptive Execution Plans

Displaying Adaptive Plans (+report)

```
Adaptive plan:
--------------
This cursor has an adaptive plan, but adaptive plans are enabled for
reporting mode only.  The plan that would be executed if adaptive plans
were enabled is displayed below.

Plan hash value: 1255158658

-------------------------------------------------------------------------------------
| Id  | Operation           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                     |       |       |     7 (100)|          |
|*  1 |  HASH JOIN          |                     |     4 |   128 |     7   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| ORDER_ITEMS         |     4 |    48 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| PRODUCT_INFORMATION |     1 |    20 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

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: 1553478007

-------------------------------------------------------------------------------------
| Id  | Operation           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                     |     1 |    32 |     8   (0)| 00:00:01 |
|*  1 |  HASH JOIN          |                     |     1 |    32 |     8   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| ORDER_ITEMS         |    13 |   156 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| PRODUCT_INFORMATION |   288 |  5760 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

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
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!

enkitec

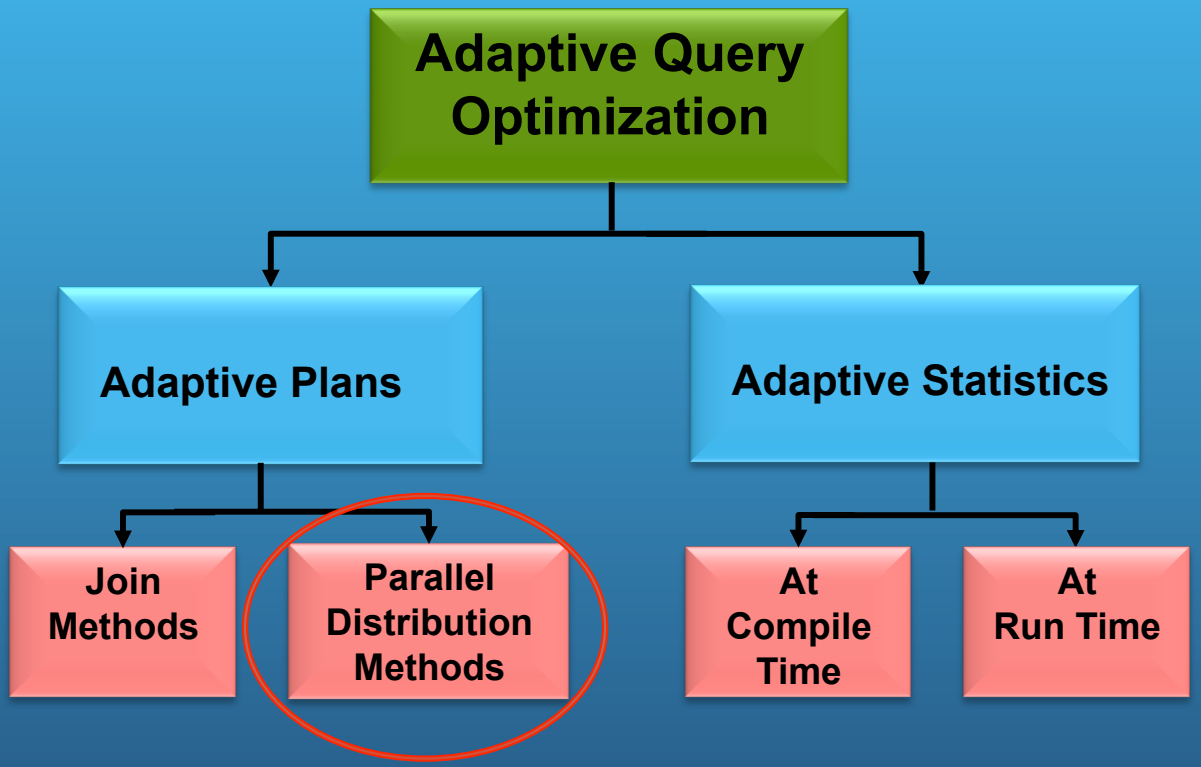
Adaptive Execution Plans

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

-------------------------------------------------------------------------------------------
| Id  | Operation           | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |                     |       |       |     8 (100)|          |
|*  1 |  HASH JOIN          |                     |    13 |   416 |     8   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| ORDER_ITEMS         |    13 |   156 |     3   (0)| 00:00:01 |
|   3 |   TABLE ACCESS FULL| PRODUCT_INFORMATION |   288 |  5760 |     5   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------

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



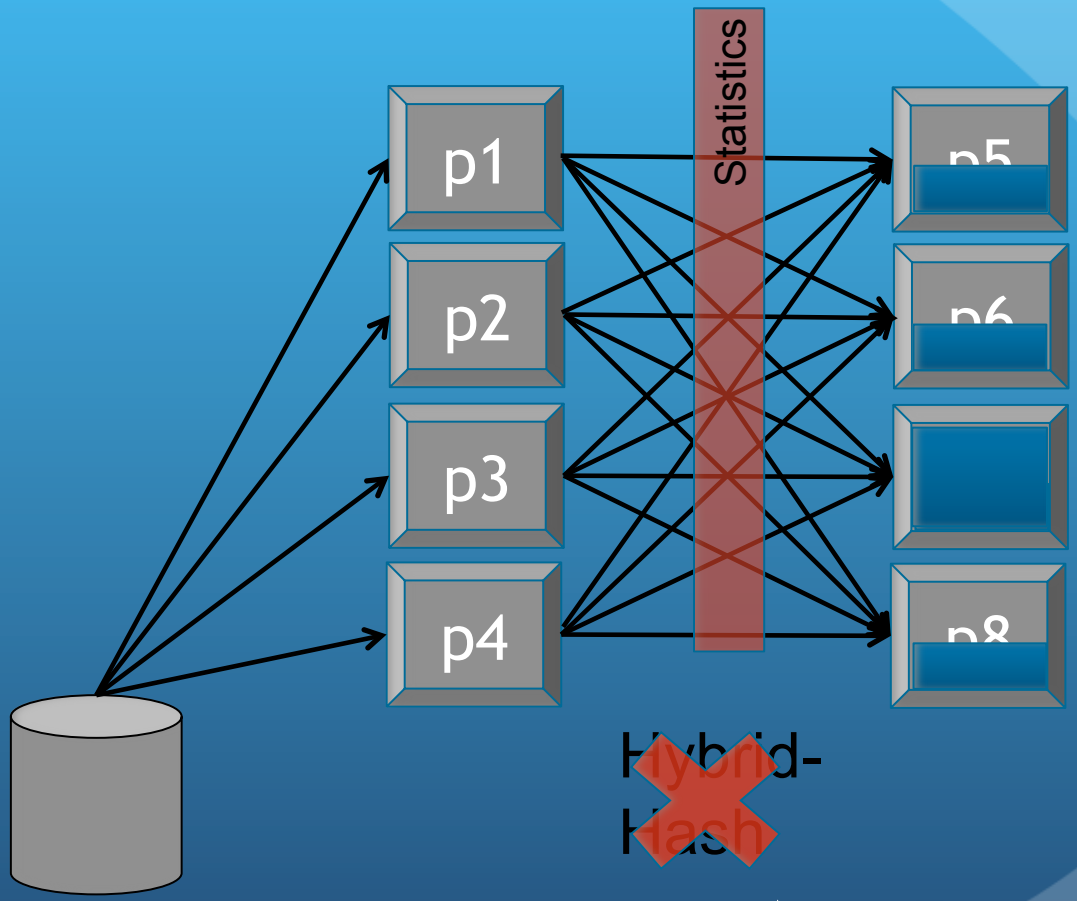
enkitec

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

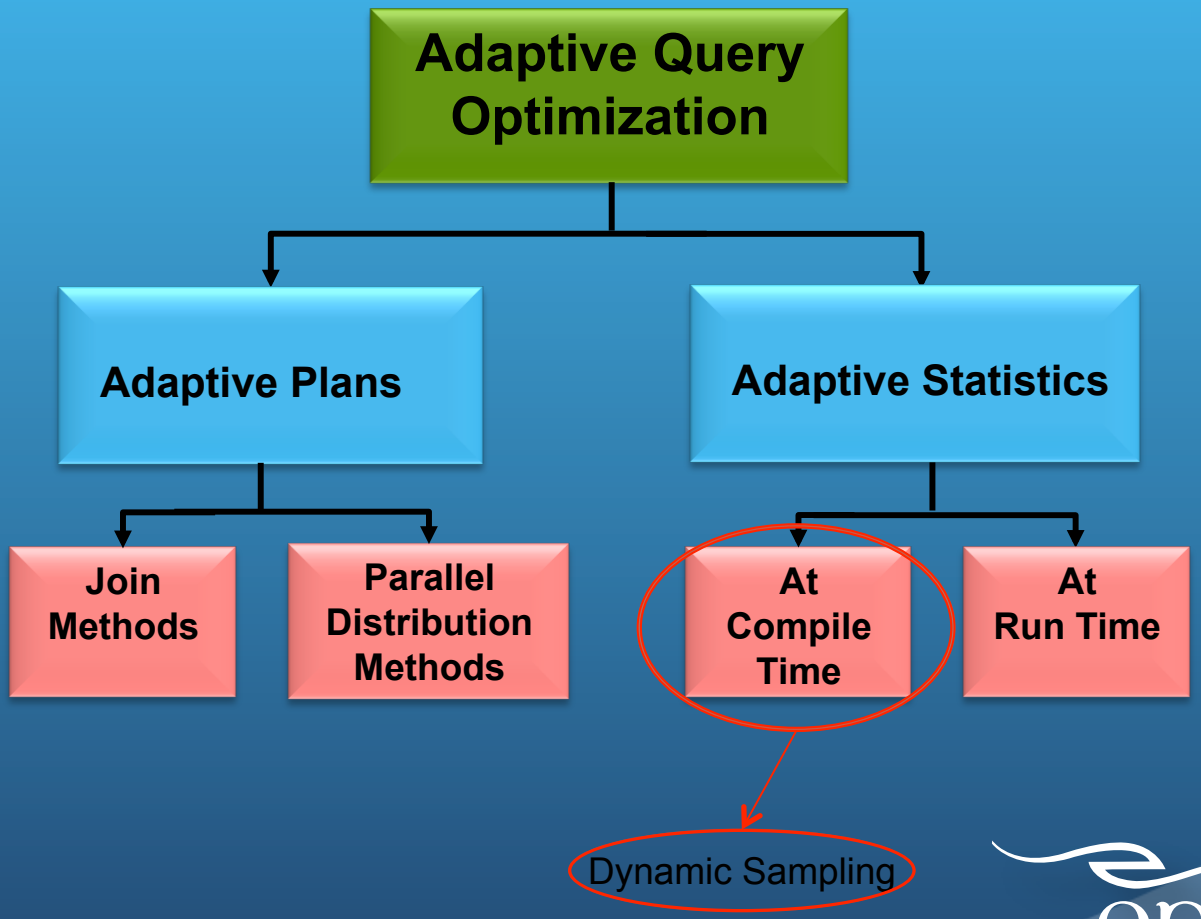
Adaptive Distribution Methods

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



Adaptive Statistics

Dynamic Statistics (Sampling)



enkitec

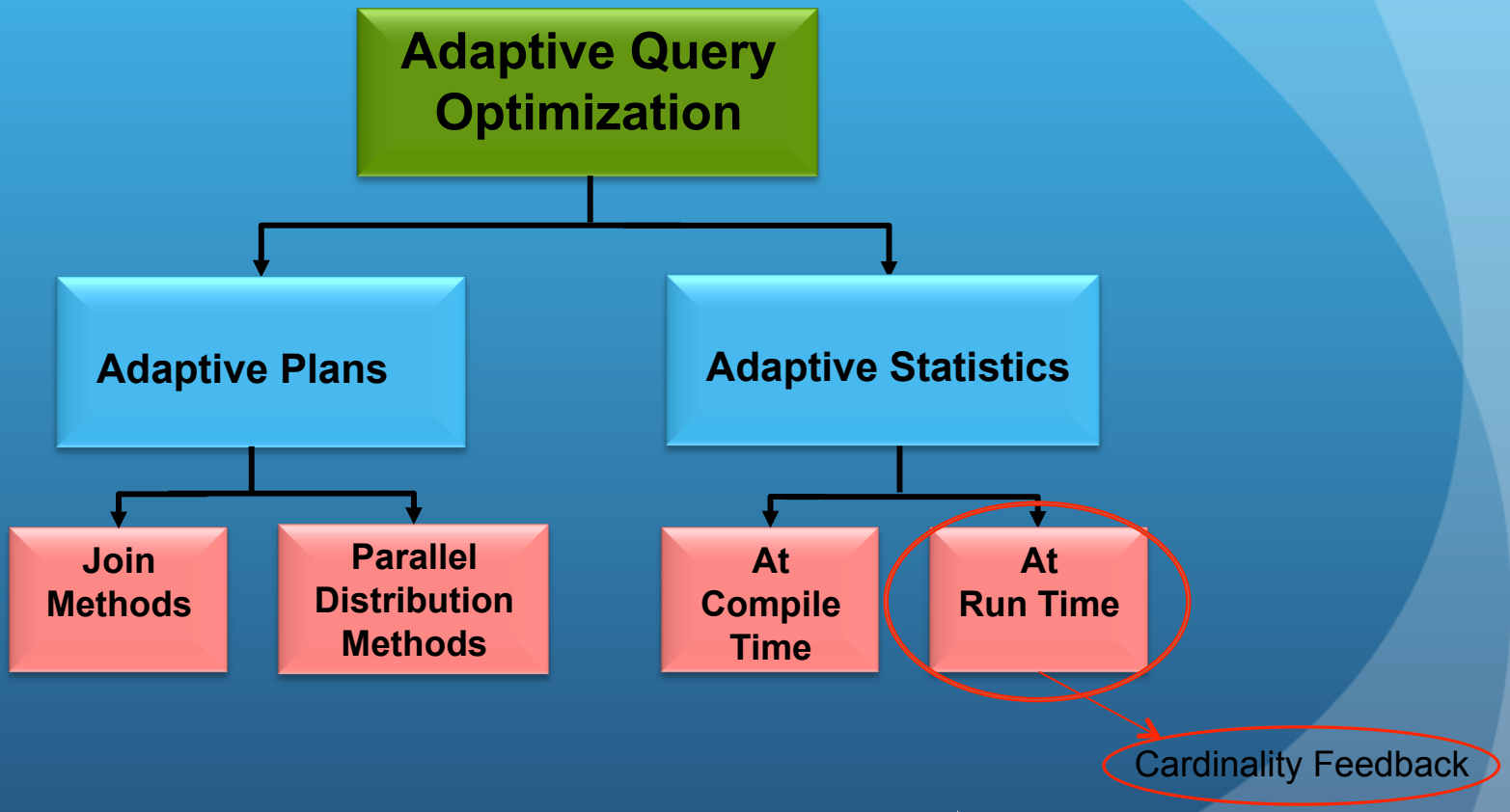
Dynamic Statistics

- 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

enkitec

Adaptive Statistics

Dynamic Statistics (Sampling)



enkitec

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

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

```
SYS@BETA3> select column_name, comments from dba_col_comments where table_name = 'DBA_SQL_PLAN_DIRECTIVES';

COLUMN_NAME                    COMMENTS
------------------------------ --------------------------------------------------------------------------------
DIRECTIVE_ID                   The identifier of the sql plan directive
TYPE                           The type of the sql plan directive
STATE                          The state of the sql plan directive
AUTO_DROP                      If YES, the sql plan directive gets dropped when unused beyond SPD_RETENTION_WEEKS
REASON                         The reason for creating the sql plan directive
CREATED                        The creation timestamp of the sql plan directive
LAST_MODIFIED                  The timestamp of most recent modification of the sql plan directive
LAST_USED                      The timestamp of most recent usage of the sql plan directive

8 rows selected.
```

```
SYS@BETA3> select column_name, comments from dba_col_comments where table_name = 'DBA_SQL_PLAN_DIR_OBJECTS';

COLUMN_NAME                    COMMENTS
------------------------------ --------------------------------------------------------------------------------
DIRECTIVE_ID                   The identifier of the sql plan directive
OWNER                          The username of the owner of the object in the sql plan directive
OBJECT_NAME                    The name of the object in the sql plan directive
SUBOBJECT_NAME                 The name of the sub-object (for example column) in the sql plan directive
OBJECT_TYPE                    The type of the (sub-)object in the sql plan directive
NOTES                          Other notes about the object
```

SQL Plan Directives

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

            DIRECTIVE_ID OWNER      OBJECT_NAME                    SUBOBJECT_NAME                 OBJECT
------------------------ ---------- ------------------------------ ------------------------------ ------
    14460712757220495343 OE         ORDER_ITEMS                    UNIT_PRICE                     COLUMN
                         OE         ORDER_ITEMS                    QUANTITY                       COLUMN
                         OE         ORDER_ITEMS                                                   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

           DIRECTIVE_ID TYPE             STATE         REASON                                  CREATED
----------------------- ---------------- ------------- --------------------------------------- ------------------------------
   14460712757220495343 DYNAMIC_SAMPLING HAS_STATS     SINGLE TABLE CARDINALITY MISESTIMATE 04-MAR-13 11.15.38.000000 PM

           DIRECTIVE_ID OWNER      OBJECT_NAME                    NOTES
----------------------- ---------- ------------------------------ --------------------------------------------------------------------------------
   14460712757220495343 OE         ORDER_ITEMS                    <obj_note>
                                                                    <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>
```

enkitec

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

           DIRECTIVE_ID TYPE            STATE        REASON                                  CREATED
----------------------- --------------- ------------ --------------------------------------- ------------------------------
    14460712757220495343 DYNAMIC_SAMPLING HAS_STATS    SINGLE TABLE CARDINALITY MISESTIMATE 04-MAR-13 11.15.38.000000 PM
```

SQL Plan Directives

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

TYPE             REASON                                   STATE
---------------- ---------------------------------------- -------------
DYNAMIC_SAMPLING GROUP BY CARDINALITY MISESTIMATE          HAS_STATS
                                                          NEW

                 JOIN CARDINALITY MISESTIMATE             HAS_STATS
                                                          NEW
                                                          PERMANENT

                 SINGLE TABLE CARDINALITY MISESTIMATE     HAS_STATS
                                                          MISSING_STATS
                                                          NEW
                                                          PERMANENT
```

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

STATE            COUNT(*)
------------- ------------
PERMANENT              38
MISSING_STATS           7
HAS_STATS              68
NEW                    49
```

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)

enkitec

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

enkitec

Well, How Did We Get Here?

The New Optimizer

enkitec

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

enkitec

Questions?

Contact Information : Kerry Osborne
kerry.osborne@enkitec.com
kerryosborne.oracle-guy.com
www.enkitec.com