Evolution of performance management Oracle 12c adaptive optimization

Nelson Calero

OTN Tour Latinoamérica

Agosto 2016

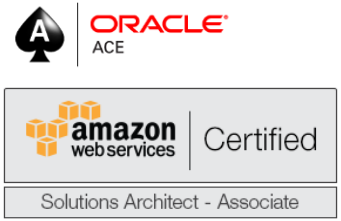
Pythian®

love your data

About me

- Database Consultant at Pythian
- Working with Oracle tools and Linux environments since 1996
- DBA Oracle (2001) & MySQL (2005)
- Co-founder and President of the Oracle user Group of Uruguay (2009)
- LAOUC Director of events (2013)

- Computer Engineer
- Oracle ACE (2014)
- Oracle Certified Professional DBA 10g/11g (2008)
- Amazon Solutions Architect – Associate since (2016)
- Oracle University Instructor (2011)
- Blogger and speaker: Oracle Open World, Collaborate, OTN Tour, Regional conferences



http://www.linkedin.com/in/ncalero

@ncalerouy

Pythian overview

- 19 Years of data infrastructure management consulting
- 250+ Top brands
- 11700+ Systems under management
- Over 400 DBAs in 35 countries
- Top 5% of DBA work force, 10 Oracle ACEs, 4 ACED,
  3 OakTable members, 2 OCM, 6 Microsoft MVPs,
  1 Cloudera Champion of Big Data,
  AWS Certified Solutions Architect – 2 Professional, 12 Associate
- Oracle, Microsoft, MySQL, Hadoop, Cassandra, MongoDB, and more
- Infrastructure, Cloud, DevOps, and application expertise

ORACLE Platinum Partner

Microsoft Partner
Silver Data Platform
Gold Data Platform

MySQL Enterprise
Authorized Platinum
REMOTE DBA SERVICES PARTNER

Pythian
love your data

Adaptive optimizations

What

- Starting in version 12.1, Oracle optimizer can adjust SQL execution plan during its first execution to create a better performing plan, and use new techniques to do better cardinality estimates on following executions

Why

- These features are enabled by default, controlled by parameters

How

- Adaptive statistics features has no extra cost, available on all editions
- Adaptive plans is available only on Enterprise edition

Adaptive optimizations

What

There are several cases where the execution plan generated by the optimizer is not the optimal for the underlying tables characteristics.

Why

This can be caused by a variety of reasons:

- Structural objects changes (datatype, indexes, partitions)
- System growth (skewed data, concurrency)
- And many more. This is a good collection of reason: http://jonathanlewis.wordpress.com/2013/12/23/plan-changes/

How

Oracle Database already have several ways to control plan execution that works at different stages in the cost based Optimizer:

- Stored Outlines (deprecated in 11.1)
- SQL Hints
- SQL Patches
- SQL Profiles
- SQL Plan Management

Adaptive optimizations

What

Why

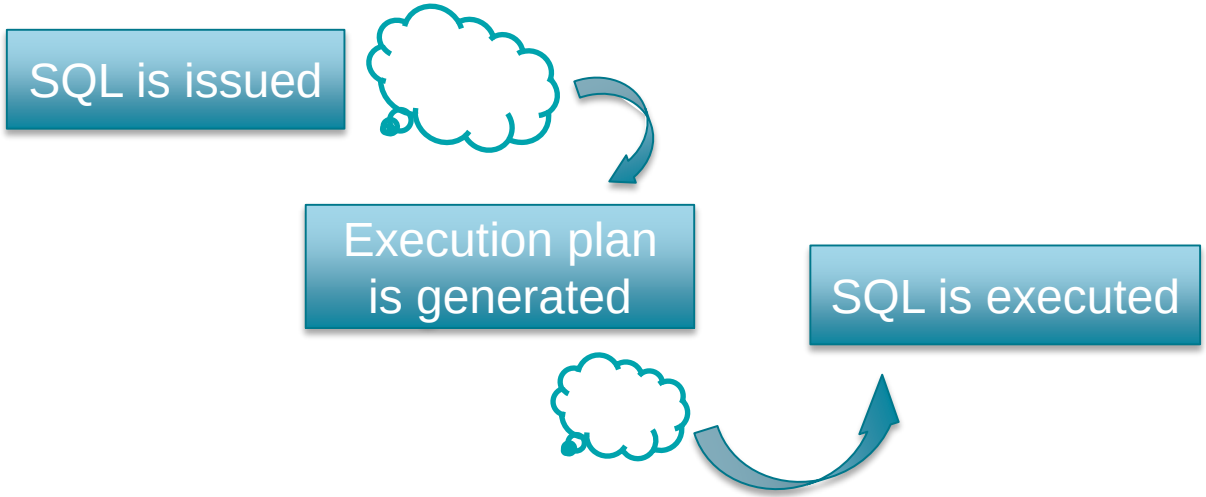
How

- Several new optimization techniques **at query run-time** – after execution plan has been chosen
- Adaptive features are evaluated only the first time a query is parsed. Next executions re-use the adaptive plan
- Some features are persisted on SYSAUX tablespace (plan directives)
- Baselines are always the last step – no adaptive optimization over a baseline plan used
- Initialization parameters and hints to control them

Adaptive optimizations - How

Oracle Cost Based Optimizer (CBO) **Before 12g**

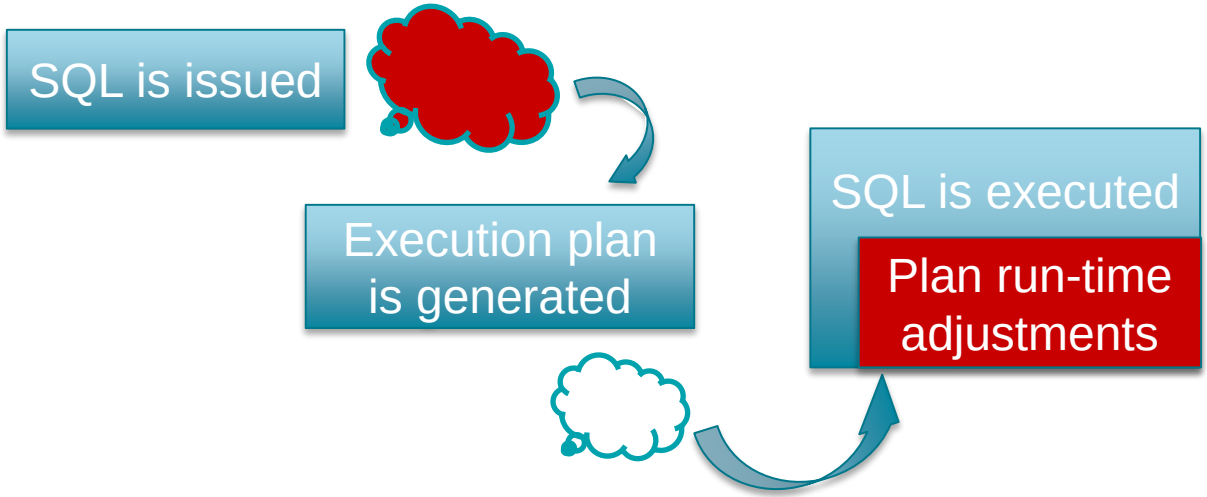
Old history



Adaptive optimizations - How

Execution plan can change during SQL execution

Oracle Cost Based Optimizer (CBO) **Since 12g**



Review of Adaptive optimizations features

- Adaptive statistics
  - Dynamic Statistics
  - Automatic Reoptimization
  - SQL Plan Directives
- Adaptive plans
  - Join Methods
  - Parallel Distribution Methods

Adaptive statistics - dynamic statistics

- Was Dynamic sampling feature introduced on 10g
  - stats can be gathered for objects where not enough stats are present during **parsing**
  - Triggered because of missing or stale stats or complex predicates
  - Level (0-10) control when it fires and blocks size of samples (default=2)
- New in 12c:
  - level 11: Optimizer decides if dynamic stats should be taken and level to use. Backported to 11.2.0.4
  - Support for joins and group by predicates
- Dynamic stats are stored in memory (result cache) - reused by other SQL
  - This can be seen on a SQL trace (not 10053)

Adaptive statistics - dynamic statistics

```
SCOTT@db12102/12.1.0.2>
select status, enabled, count(1)
from data
group by status, enabled
order by 1,2;

S E   COUNT(1)
- - ----------
A Y      2499
C N      5000
C Y      2500
```

1

2

```
SCOTT@db12102/12.1.0.2> @q
SCOTT@db12102/12.1.0.2> @q
```

```
SQL_ID      3q3tk8z3su2px, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */  count(1) FROM   data
WHERE status = 'C' and enabled='Y'

Plan hash value: 3102269256
-----------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |      1 | 00:00:00.01 |      61 |
|   1 |  SORT AGGREGATE    |      |      1 |      1 |      1 | 00:00:00.01 |      61 |
|*  2 |   TABLE ACCESS FULL| DATA |      1 |   3749 |   2500 | 00:00:00.01 |      61 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter(("ENABLED"='Y' AND "STATUS"='C'))
```

3

Adaptive statistics - dynamic statistics

```
SCOTT@db12102/12.1.0.2>
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
where sql_id='3q3tk8z3su2px';

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE   FULL_PHV BS BA RE AP      EXE
------------- ------------ --------------- ---------- -- -- -- -- ----------
3q3tk8z3su2px            0      3102269256 3774542448  N  N  N             2
```

Adaptive statistics - dynamic statistics

```
SCOTT@db12102/12.1.0.2>
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
where sql_id='3q3tk8z3su2px';

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE   FULL_PHV BS BA RE AP      EXE
------------- ------------ --------------- ---------- -- -- -- -- ----------
3q3tk8z3su2px            0      3102269256 3774542448  N  N  N             2
```

Lets run the query again with new level 11:

```
SCOTT@db12102/12.1.0.2> alter session set optimizer_dynamic_sampling=11;

Session altered.

SCOTT@db12102/12.1.0.2> SELECT /*+ GATHER_PLAN_STATISTICS */ count(1) FROM data
WHERE status = 'C' and enabled='Y';
```

Adaptive statistics - dynamic statistics

```
SQL_ID      3q3tk8z3su2px, child number 1
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */  count(1) FROM   data
WHERE status = 'C' and enabled='Y'

Plan hash value: 3102269256
-----------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |      A-Time | Buffers |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |      1 | 00:00:00.01 |      61 |
|   1 |  SORT AGGREGATE    |      |      1 |      1 |      1 | 00:00:00.01 |      61 |
|*  2 |   TABLE ACCESS FULL| DATA |      1 |   2500 |   2500 | 00:00:00.01 |      61 |
-----------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter(("ENABLED"='Y' AND "STATUS"='C'))

Note
-----
   - dynamic statistics used: dynamic sampling (level=AUTO)
```

Adaptive statistics - dynamic statistics

```
SCOTT@db12102/12.1.0.2>
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
where sql_id='3q3tk8z3su2px';

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE   FULL_PHV BS BA RE AP      EXE
------------- ------------ --------------- ---------- -- -- -- -- ----------
3q3tk8z3su2px            0      3102269256 3774542448 N  N  N              2
3q3tk8z3su2px            1      3102269256 3774542448 N  N  N              2
```

New cursor created because of dynamic sampling

dynamic statistics – SQL trace (10053)

Access path analysis for TAB1

****

SINGLE TABLE ACCESS PATH

Single Table Cardinality Estimation for TAB1[TAB1]

SPD: Directive valid: dirid = 3423856709119923569, state = 1, flags = 1, loc = 1 {EC(92667)[3, 4, 5]}

SPD: Return code in qosdDSDirSetup: EXISTS, estType = TABLE

...

Table: TAB1 Alias: TAB1

Card: Original: 10000.000000 >> Single Tab Card adjusted from 1869.000000 to 5000.000000 due to adaptive dynamic sampling

Rounded: 5000 Computed: 5000.000000 Non Adjusted: 1869.000000

Review of Adaptive optimizations features

- Adaptive statistics
  - ✓ Dynamic Statistics
  - **Automatic Reoptimization**
  - SQL Plan Directives
- Adaptive plans
  - Join Methods
  - Parallel Distribution Methods

Adaptive statistics - Automatic reoptimization

Stats from previous query execution is used to create a better plan

1) statistics feedback
   - was cardinality feedback introduced on 11.2
     - when SQL has no stats, multiple predicates or where no accurate selectivity can be computed
     - after execution, cardinality estimates are compared to actual values on each operation, and differences stored for use on next execution
     - stored in memory by statement
   - SQL is marked as reoptimizable – v$sql.is_reoptimizable
   - Join stats are compared
   - Plan directive can be created to persist this information

Adaptive statistics – statistics feedback

```
SCOTT@db12102/12.1.0.2>
select status, enabled, count(1)
from data
group by status, enabled
order by 1,2;

S E   COUNT(1)
- - ----------
B Y      2499
C N      5000
C Y      2500
```

1

2

```
SCOTT@db12102/12.1.0.2> @q2
SCOTT@db12102/12.1.0.2> @q2
```

Similar query, now selecting all data instead of count(*)

```
SQL_ID      0h00c8djb1f0h, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ * FROM   data
WHERE  status = 'B' and enabled='Y'

Plan hash value: 3160396028
-----------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |     A-Time | Buffers |
-----------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |   2499 |00:00:00.01 |     228 |
|*  1 |  TABLE ACCESS FULL| DATA |      1 |   1249 |   2499 |00:00:00.01 |     228 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(("STATUS"='B' AND "ENABLED"='Y'))
```

3

Adaptive statistics – statistics feedback

```
SQL_ID      0h00c8djb1f0h, child number 1
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ * FROM   data
WHERE  status = 'B' and enabled='Y'

Plan hash value: 3160396028
-------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |     A-Time | Buffers |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |   2499 |00:00:00.01 |     228 |
|*  1 |  TABLE ACCESS FULL| DATA |      1 |   2499 |   2499 |00:00:00.01 |     228 |
-------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(("STATUS"='B' AND "ENABLED"='Y'))

Note
-----
   - statistics feedback used for this statement
```

Executed the query again

Adaptive statistics – statistics feedback

```
SCOTT@db12102/12.1.0.2>
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
where sql_id='0h00c8djb1f0h';

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE   FULL_PHV BS BA RE AP      EXE
------------- ------------ --------------- ---------- -- -- -- -- ----------
0h00c8djb1f0h            0      3160396028 2510600540 N  N  Y             1
0h00c8djb1f0h            1      3160396028 2510600540 N  N  N             1
```

New cursor created because of statistics feedback

Adaptive statistics - Automatic reoptimization

2) Performance feedback

- new ADAPTIVE value for PARALLEL_DEGREE_POLICY
- the degree of parallelism (DOP) is automatically chosen as in AUTO mode, but at the end actual performance is compared with estimated, which can lead to a reoptimization on next execution (cursor marked as reoptimizable and statistics feedback stored for reuse)

Review of Adaptive optimizations features

- Adaptive statistics
  - ✓ Dynamic Statistics
  - ✓ Automatic Reoptimization
  - **SQL Plan Directives**
- Adaptive plans
  - Join Methods
  - Parallel Distribution Methods

Adaptive statistics – SQL plan directives

- Stored feedback from SQL execution to adjust statistics on query expressions for next execution, created by automatic reoptimization
  - not in all cases (previous example about statistics)
- Stored in memory (result cache), flushed to SYSAUX each 15 min.
  - Purge policy in place
  - DBMS_SPD package to maintain them
  - Cannot be manually created
- DBA_SQL_PLAN_* views

⇒ All you want to know about it: UKOUG15 Frank Pachot session “SQL Plan Directives - The Memory of the 12c Optimizer”

Adaptive statistics – SQL plan directives

```
SYS@db12102/12.1.0.2> SELECT d.type, d.state, d.ENABLED,
       count(distinct d.directive_id) cant_dir, reason
FROM   dba_sql_plan_directives d, dba_sql_plan_dir_objects o
WHERE  d.directive_id=o.directive_idgroup by d.type, d.state, d.ENABLED, reason
ORDER BY 1,2,3;

TYPE             STATE      ENA CANT_DIR    REASON
---------------- ---------- --- ---------- --------------------------------
DYNAMIC_SAMPLING SUPERSEDED YES          2 GROUP BY CARDINALITY MISESTIMATE
DYNAMIC_SAMPLING SUPERSEDED YES         15 JOIN CARDINALITY MISESTIMATE
DYNAMIC_SAMPLING SUPERSEDED YES         14 SINGLE TABLE CARDINALITY MISESTIMATE
DYNAMIC_SAMPLING USABLE     YES         22 JOIN CARDINALITY MISESTIMATE
DYNAMIC_SAMPLING USABLE     YES         24 SINGLE TABLE CARDINALITY MISESTIMATE
```

Adaptive statistics – SQL plan directives

```
SYS@db12102/12.1.0.2> SELECT o.owner, d.type, d.state, d.ENABLED
          ,count(*) cant, count(distinct d.directive_id) cant_dir
          ,count(distinct object_name) cant_obj
FROM   dba_sql_plan_directives d, dba_sql_plan_dir_objects o
WHERE  d.directive_id=o.directive_idgroup by o.owner, d.type, d.state, d.ENABLED
ORDER BY 1,2,3;

OWNER      TYPE             STATE      ENA       CANT   CANT_DIR   CANT_OBJ
---------- ---------------- ---------- --- ---------- ---------- ----------
CTXSYS     DYNAMIC_SAMPLING USABLE     YES          1          1          1
DBSNMP     DYNAMIC_SAMPLING SUPERSEDED YES          2          2          1
SYS        DYNAMIC_SAMPLING SUPERSEDED YES         93         31         43
SYS        DYNAMIC_SAMPLING USABLE     YES        116         46         36
```

Review of Adaptive optimizations features

- Adaptive statistics
  - ✓ Dynamic Statistics
  - ✓ Automatic Reoptimization
  - ✓ SQL Plan Directives
- **Adaptive plans**
  - Join Methods
  - Parallel Distribution Methods

Adaptive plans

- Subplans are created at parse time and decision to choose one is deferred until actual execution. A default plan is created
- A new step in the execution plan monitors rows processed (collector) to validate if predicted cardinality is accurate
- A subplan is chosen, default plan can change:
  - join method (between Nested Loop (NL) and Hash Join (HJ))
  - parallel distribution method
- New adaptive plan is reused on future executions, no more optimization are repeated

Adaptive plans – example

```
SCOTT@db12102/12.1.0.2> select table_name, num_rows
from user_tables where table_name in ('T1','T2');

TABLE_NAME             NUM_ROWS
-------------------- ----------
T1                        91138
T2                           99

SCOTT@db12102/12.1.0.2> insert into t2
 select rownum+100, mod(rownum,10) type, object_name data
 from dba_objects where rownum < 1000;

 999 rows created.

 SCOTT@db12102/12.1.0.2> alter system flush shared_pool;
```

Lets see how a query is resolved without any adaptive optimization

```
SCOTT@db12102/12.1.0.2> alter session set optimizer_adaptive_features=false;

Session altered.
```

Adaptive plans – example

```
SQL_ID      5bvpk0nnd4323, child number 1
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --, t2.data FROM  t2, t1 WHERE t1.id = t2.id2

Plan hash value: 1142304008

-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name  | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |       |      1 |        |   1098 |00:00:00.03 |    1291 |
|   1 |  NESTED LOOPS                 |       |      1 |     99 |   1098 |00:00:00.03 |    1291 |
|   2 |   NESTED LOOPS                |       |      1 |     99 |   1098 |00:00:00.03 |     193 |
|   3 |    INDEX FULL SCAN            | T2_PK |      1 |     99 |   1098 |00:00:00.01 |      81 |
|*  4 |    INDEX UNIQUE SCAN          | T1_PK |   1098 |      1 |   1098 |00:00:00.01 |     112 |
|   5 |   TABLE ACCESS BY INDEX ROWID| T1    |   1098 |      1 |   1098 |00:00:00.01 |    1098 |
-----------------------------------------------------------------------------------------------------

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

   4 - access("T1"."ID"="T2"."ID2")

23 rows selected.
```

Big difference in estimated rows

Nested Loop is used

Adaptive plans – example

Same query now with adaptive optimization enabled:

```
SCOTT@db12102/12.1.0.2> alter session set optimizer_adaptive_features=true;

Session altered.

Elapsed: 00:00:00.00

SCOTT@db12102/12.1.0.2> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.04
```

Adaptive plans – example

```
SQL_ID      5bvpk0nnd4323, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */   t1.id, t1.data --, t2.data FROM  t2, t1 WHERE      t1.id = t2.id2

Plan hash value: 324465990

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation          | Name | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |      1 |        |   1098 |00:00:00.01 |     560 |       |       |          |
|*  1 |  HASH JOIN         |      |      1 |     99 |   1098 |00:00:00.01 |     560 |  1263K|  1263K| 1302K (0)|
|   2 |   TABLE ACCESS FULL| T2   |      1 |     99 |   1098 |00:00:00.01 |       7 |       |       |          |
|   3 |   TABLE ACCESS FULL| T1   |      1 |      1 |  91138 |00:00:00.14 |     553 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------

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

   1 - access("T1"."ID"="T2"."ID2")

Note
-----
   - this is an adaptive plan
```

estimation didn't improved
but plan has changed to HJ

Adaptive plans – example - adaptive steps

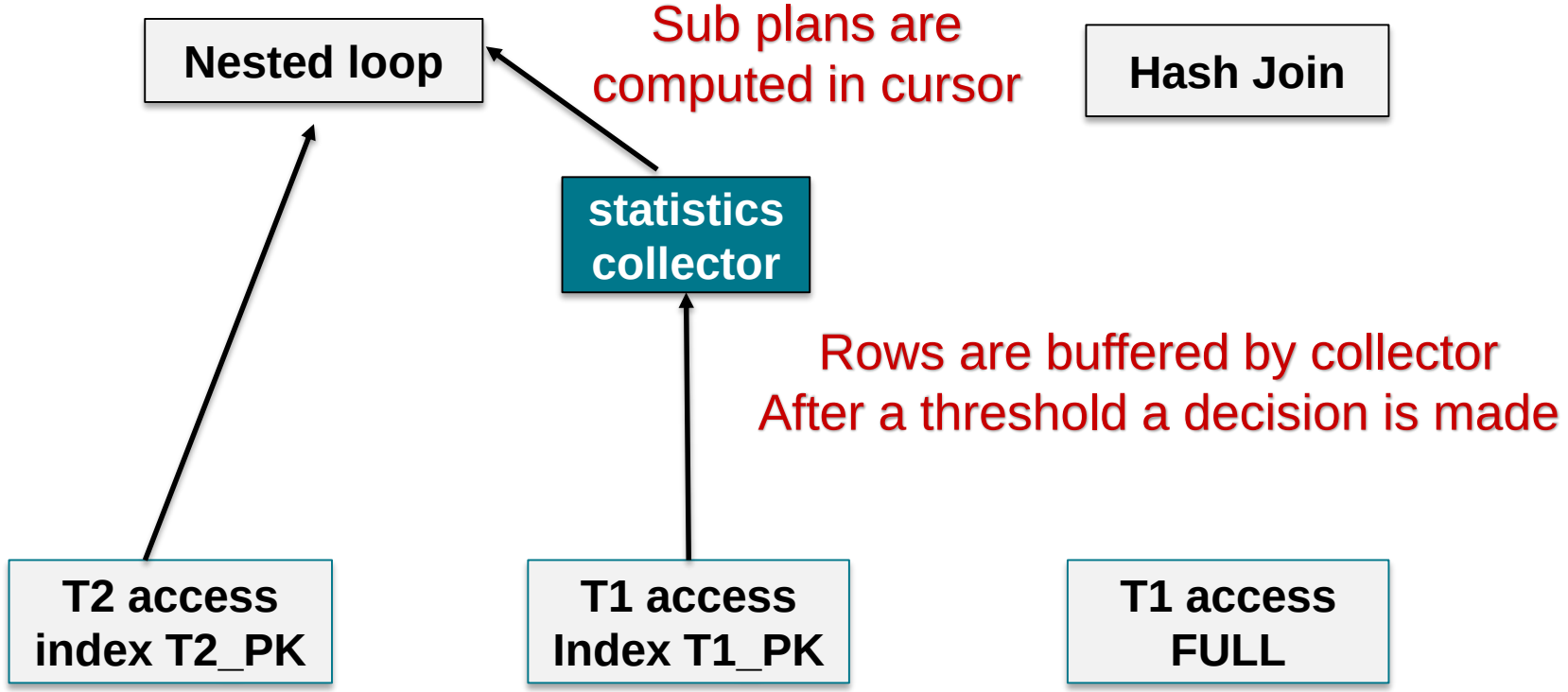
```
SQL_ID 5bvpk0nnd4323, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */    t1.id, t1.data --, t2.data FROM t2, t1 WHERE   t1.id = t2.id2

Plan hash value: 324465990
```

	Id	Operation	Name	Starts	E-Rows	A-Rows	A-Time	Buffers	OMem	1Mem	Used-Mem
	0	SELECT STATEMENT		1		1098	00:00:00.01	560			
*	1	HASH JOIN		1	99	1098	00:00:00.01	560	1263K	1263K	1302K (0)
-	2	NESTED LOOPS		1	99	1098	00:00:00.01	7			
-	3	NESTED LOOPS		1	99	1098	00:00:00.01	7			
-	4	STATISTICS COLLECTOR		1		1098	00:00:00.01	7			
	5	INDEX FULL SCAN	T2_PK	1	99	1098	00:00:00.01	7			
-	* 6	INDEX UNIQUE SCAN	T1_PK	0	1	0	00:00:00.01	0			
-	7	TABLE ACCESS BY INDEX ROWID	T1	0	1	0	00:00:00.01	0			
	8	TABLE ACCESS FULL	T1	1	1	91138	00:00:00.14	553			

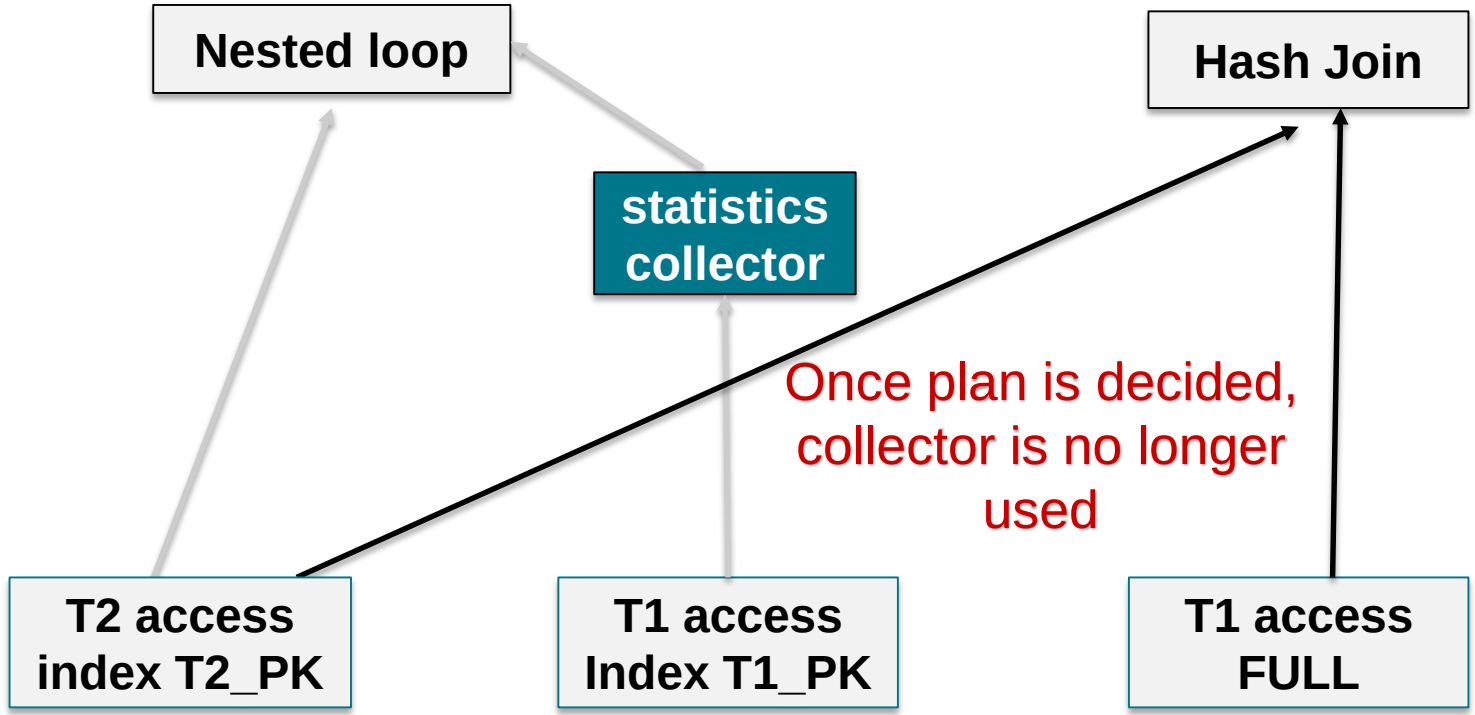
```
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("T1"."ID"="T2"."ID2")
   6 - access("T1"."ID"="T2"."ID2")

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

dbms_xplan.display_cursor(format=>'adaptive')

Adaptive plans – statistics collector step



Adaptive plans – statistics collector step



Adaptive plans – extra work?

- Sub plans are parsed
- During first execution, plan change can be decided. It does not restart the query, as rows already processed are used
- Following executions use the generated plan, no adaptive optimization is performed

```
SELECT sql_id, child_number, is_bind_sensitive BS,
       is_bind_aware BA, IS_REOPTIMIZABLE RE,
       IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
WHERE sql_id='5bvpk0nnd4323';

SQL_ID        CHILD_NUMBER BS BA RE AP EXE
------------- ------------ -- -- -- -- ---
5bvpk0nnd4323            0  N  N  N  Y   5
```

Adaptive plans – nested loop instead of HJ

```
SQL_ID 5bvpk0nnd4323, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */  t1.id, t1.data --, t2.data FROM   t1, t2 WHERE  t1.id = t2.id2

Plan hash value: 1142304008
-----------------------------------------------------------------------------------------------------------
|   Id  | Operation                     | Name  | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |       |      1 |        |     99 |00:00:00.01 |     115 |
|- *  1 |  HASH JOIN                    |       |      1 |     99 |     99 |00:00:00.01 |     115 |
|     2 |   NESTED LOOPS                |       |      1 |     99 |     99 |00:00:00.01 |     115 |
|     3 |    NESTED LOOPS               |       |      1 |     99 |     99 |00:00:00.01 |      16 |
|-    4 |     STATISTICS COLLECTOR      |       |      1 |        |     99 |00:00:00.01 |       1 |
|     5 |      INDEX FULL SCAN          | T2_PK |      1 |     99 |     99 |00:00:00.01 |       1 |
|  *  6 |     INDEX UNIQUE SCAN         | T1_PK |     99 |      1 |     99 |00:00:00.01 |      15 |
|     7 |    TABLE ACCESS BY INDEX ROWID| T1    |     99 |      1 |     99 |00:00:00.01 |      99 |
|-    8 |   TABLE ACCESS FULL           | T1    |      0 |      1 |      0 |00:00:00.01 |       0 |
-----------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("T1"."ID"="T2"."ID2")
   6 - access("T1"."ID"="T2"."ID2")

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

earlier HJ was used when A-rows=1098

Adaptive plans – what triggers HJ/NL?

- Subsets cardinality

- Previous examples:
  - T2 with 99 rows, updated stats -> adaptive NL
  - T2 with 1098 rows, stats saying 99 rows -> adaptive HJ

- Threshold used by collector = inflection point
  - It can be seen on 10053 trace

Adaptive plans – optimizer trace (10053)

```
SQL> alter session set tracefile_identifier='adjhj2';
SQL> exec dbms_sqldiag.dump_trace(p_sql_id=>'9vbxajh8hsng8',p_child_number=>0,
         p_component=>'Compiler',p_file_id=>'');

SQL> select tracefile from v$process
     where addr = (select paddr from v$session where sid = userenv('sid'));
TRACEFILE
------------------------------------------------------------------------------
/u01/app/oracle/diag/rdbms/db12102/db12102/trace/db12102_ora_2558_adjhj2.trc

[oracle@bigdatalite ~]$ grep -c inflection
/u01/app/oracle/diag/rdbms/db12102/db12102/trace/db12102_ora_2558_adjhj2.trc
67
[oracle@bigdatalite ~]$ grep inflection
/u01/app/oracle/diag/rdbms/db12102/db12102/trace/db12102_ora_2558_adjhj2.trc | tail
AP: Costing Nested Loops Join for inflection point at card 134.68
AP: Costing Hash Join for inflection point at card 134.68
AP: Searching for inflection point at value 134.68
AP: Costing Nested Loops Join for inflection point at card 135.30
AP: Costing Hash Join for inflection point at card 135.30
AP: Costing Hash Join for inflection point at card 135.30
DP: Found point of inflection for NLJ vs. HJ: card = 135.30
```

Adaptive plans – keeping track of plan changes

PLAN_HASH_VALUE is still useful?

We need to look after FULL_PLAN_HASH_VALUE

```
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
WHERE sql_id='5bvpk0nnd4323';
```

NOTE: output from different runs

SQL_ID	CHILD_NUMBER	PLAN_HASH_VALUE	FULL_PHV	BS	BA	RE	AP	EXE	
5bvpk0nnd4323	0	**1142304008**	**1330942548**	N	N	N	Y	2	← **NL**
5bvpk0nnd4323	0	**324465990**	**1330942548**	N	N	N	Y	1	← **HJ**
5bvpk0nnd4323	1	**1142304008**	830081733	N	N	N		2	← NL no adaptive

Adaptive plans – displaying plan

- dbms_xplan.display_cursor - available formats:
  - **format => 'adaptive'**
    - always accurate
    - without it, final plan is shown not including discarded steps.
    - add 'allstats last' to see estimated and actual rows
  - **format => 'report'**
    - only works when OPTIMIZER_ADAPTIVE_REPORTING_ONLY=TRUE
    - display what could have been the plan if adaptive features were enabled

```
select * from table(dbms_xplan.display_cursor(format => 'adaptive allstats last'));
```

- autotrace: does not show the final plan

- dbms_xplan.display_sql_plan_baseline
  - not always showing the adaptive steps, or the correct note section

SPM (baselines) with adaptive plan

From docs:

- final plan used is captured as baseline if using automatic capture
- when there is a baseline and a new adaptive plan appears, the default plan is captured and marked as adaptive
- accepted plans are never adaptive
- evolution of an adaptive plan accepts the real used plan after evaluating all possible new plans, deleting the old adaptive one

$\Rightarrow$ Lets test this using same queries from previous examples

SPM (baselines) with adaptive plan

```
@test-join-nl  -- only 99 rows on T2

alter session set
  optimizer_capture_sql_plan_baselines = TRUE;

-- using same SQL:
SELECT /*+ GATHER_PLAN_STATISTICS */
       t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

/         -- executed two times

alter session set
 optimizer_capture_sql_plan_baselines = FALSE;
```

```
col signature for 99999999999999999999
col sql_handle for a20
col plan_name for a30

select signature, sql_handle, plan_name, enabled,
       accepted, fixed, adaptive, sql_text
from dba_sql_plan_baselines;

SIGNATURE             SQL_HANDLE
PLAN_NAME                      ENA ACC FIX ADA
SQL_TEXT
--------------------- --------------------
------------------------------ --- --- --- ---
--------------------------------------------------------------------------------
 12148921326023345300 SQL_a899a39409ca0494
SQL_PLAN_aj6d3kh4wn14n317a0ac5 YES YES NO  NO
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --,
t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2
```

SPM (baselines) with adaptive plan

```
SQL_ID      5bvpk0nnd4323, child number 3
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data --, t2.data FROM t2, t1 WHERE  t1.id = t2.id2
Plan hash value: 1142304008
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name  | Starts | E-Rows | A-Rows |   A-Time    | Buffers  |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |       |      1 |        |     99 |00:00:00.01 |      122 |
|   1 |  NESTED LOOPS                |       |      1 |     99 |     99 |00:00:00.01 |      122 |
|   2 |   NESTED LOOPS               |       |      1 |     99 |     99 |00:00:00.01 |       23 |
|   3 |    INDEX FULL SCAN           | T2_PK |      1 |     99 |     99 |00:00:00.01 |        8 |
|*  4 |    INDEX UNIQUE SCAN         | T1_PK |     99 |      1 |     99 |00:00:00.01 |       15 |
|   5 |   TABLE ACCESS BY INDEX ROWID| T1    |     99 |      1 |     99 |00:00:00.01 |       99 |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   4 - access("T1"."ID"="T2"."ID2")
Note
-----
   - SQL plan baseline SQL_PLAN_aj6d3kh4wn14n317a0ac5 used for this statement
```

After query is executed
same plan and baseline is used

SPM (baselines) with adaptive plan

```
SELECT sql_id, child_number, plan_hash_value, full_plan_hash_value full_phv, SQL_PLAN_BASELINE,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP, executions exe
FROM   v$sql
WHERE sql_id='5bvpk0nnd4323';

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE FULL_PHV   SQL_PLAN_BASELINE                BS BA RE AP  EXE
------------- ------------ --------------- ---------- ------------------------------ -- -- -- -- ----
5bvpk0nnd4323            0      1142304008 1330942548                                N  N  N  Y    1
5bvpk0nnd4323            1      1142304008 1330942548                                N  N  N  Y    2
5bvpk0nnd4323            3      1142304008  830081733 SQL_PLAN_aj6d3kh4wn14n317a0ac5 N  N  N       2
```

⇒ Baseline plan is the same as adaptive but FULL_PLAN_HASH_VALUE is different

- FPHV **830081733** is the non-adaptive seen earlier

SPM (baselines) with adaptive plan

What if we add more data to T2 as we did before, plan changes to HJ?

⇒ No, baseline is used and no new plan is captured

If the query uses bind variables, plan changes to HJ?

⇒ now the adaptive plan appears as a non accepted baseline

```
var t number;
exec :t := 2;
SELECT …
WHERE  t1.id = t2.id2 and t1.type=:t;
```

SPM (baselines) with adaptive plan - binds

This is the query with binds when it runs without any baseline, T2 with 1098 rows

```
SQL_ID a54t8xnmpcqza, child number 0
-------------------------------------
SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data, t2.data FROM   t2,t1 WHERE  t1.id = t2.id2 and t1.type=:t

Plan hash value: 2959412835
-----------------------------------------------------------------------------------------------------------------------------------------
|   Id  | Operation                     | Name  | Starts | E-Rows | A-Rows |   A-Time    | Buffers |  OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |       |      1 |        |     11 |00:00:00.01 |     488 |       |       |          |
|  *  1 |  HASH JOIN                    |       |      1 |     99 |     11 |00:00:00.01 |     488 |  1263K|  1263K| 1286K (0)|
|-    2 |   NESTED LOOPS                |       |      1 |     99 |   1098 |00:00:00.01 |       7 |       |       |          |
|-    3 |    NESTED LOOPS               |       |      1 |     99 |   1098 |00:00:00.01 |       7 |       |       |          |
|-    4 |     STATISTICS COLLECTOR      |       |      1 |        |   1098 |00:00:00.01 |       7 |       |       |          |
|     5 |      TABLE ACCESS FULL        | T2    |      1 |     99 |   1098 |00:00:00.01 |       7 |       |       |          |
|- *  6 |     INDEX UNIQUE SCAN         | T1_PK |      0 |      1 |      0 |00:00:00.01 |       0 |       |       |          |
|- *  7 |    TABLE ACCESS BY INDEX ROWID| T1    |      0 |      1 |      0 |00:00:00.01 |       0 |       |       |          |
|  *  8 |   TABLE ACCESS FULL           | T1    |      1 |      1 |    912 |00:00:00.01 |     481 |       |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------

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

SPM (baselines) with adaptive plan - binds

```
SCOTT@db12102/12.1.0.2> SELECT sql_id, child_number, exact_matching_signature, plan_hash_value,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, IS_RESOLVED_ADAPTIVE_PLAN AP
FROM   v$sql
where sql_id='a54t8xnmpcqza';

SQL_ID        CHILD_NUMBER EXACT_MATCHING_SIGNATURE PLAN_HASH_VALUE BS BA RE AP
------------- ------------ ----------------------- --------------- -- -- -- --
a54t8xnmpcqza            0      3456246857634883437      2959412835  Y  N  N  Y  ← normal exec, no baseline
a54t8xnmpcqza            1      3456246857634883437      2959412835  Y  N  N  Y  ← baseline capture is enabled
a54t8xnmpcqza            2      3456246857634883437      2959412835  N  N  N     ← no capture, baseline enabled

select signature, sql_handle, plan_name, enabled, accepted, fixed, adaptive
from dba_sql_plan_baselines;

        SIGNATURE SQL_HANDLE       PLAN_NAME                      ENA ACC FIX ADA
-------------------- -------------------- ------------------------------ --- --- --- ---
  3456246857634883437 SQL_2ff70e347d63276d SQL_PLAN_2zxsf6jyq69vdafb5d283 YES NO  NO  NO   ← new plan captured
  3456246857634883437 SQL_2ff70e347d63276d SQL_PLAN_2zxsf6jyq69vdcea8bf8c YES YES NO  NO
```

SPM (baselines) with adaptive plan - binds

```
select * from table(dbms_xplan.display_sql_plan_baseline (plan_name => SQL_PLAN_2zxsf6jyq69vdafb5d283'));

--------------------------------------------------------------------------------
Plan name: SQL_PLAN_2zxsf6jyq69vdafb5d283     Plan id: 2947928707
Enabled: YES     Fixed: NO  Accepted: NO      Origin: AUTO-CAPTURE
Plan rows: From dictionary
--------------------------------------------------------------------------------
```

New captured plan

```
---------------------------------------------------
SQL handle: SQL_2ff70e347d63276d
SQL text: SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id,
t1.data, t2.data
FROM t2, t1 WHERE   t1.id = t2.id2 and t1.type=:t
---------------------------------------------------

Plan hash value: 2384946331
-----------------------------------------------------------------------------------------------
| Id  | Operation                                | Name    | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |         |    12 |   552 |     3   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                            |         |    12 |   552 |     3   (0)| 00:00:01 |
|   2 |   NESTED LOOPS                           |         |    12 |   552 |     3   (0)| 00:00:01 |
|   3 |    TABLE ACCESS BY INDEX ROWID BATCHED| T1      |    12 |   396 |     2   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN                     | T1_IDX  |    12 |       |     1   (0)| 00:00:01 |
|*  5 |    INDEX UNIQUE SCAN                     | T2_PK   |     1 |       |     0   (0)| 00:00:01 |
|   6 |   TABLE ACCESS BY INDEX ROWID            | T2      |     1 |    13 |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   4 - access("T1"."TYPE"=:T)
   5 - access("T1"."ID"="T2"."ID2")
```

It is non adaptive

SPM (baselines) with adaptive plan - binds

```
select * from table(dbms_xplan.display_sql_plan_baseline (plan_name => 'SQL_PLAN_2zxsf6jyq69vdcea8bf8c'));

--------------------------------------------------------------------------------
Plan name: SQL_PLAN_2zxsf6jyq69vdcea8bf8c       Plan id: 3467165580
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
Plan rows: From dictionary
--------------------------------------------------------------------------------

Plan hash value: 2959412835

-----------------------------------------------------------------------------
| Id  | Operation           | Name | Rows| Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |     |       |   102 (100)|          |
*  1 |  HASH JOIN          |      |  99 |  4554 |   102   (0)| 00:00:01 |
-----------------------------------------------------------------------------

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

   1 - access("T1"."ID"="T2"."ID2")

Note
-----
   - this is an adaptive plan
```

Original plan

```
------------------------------------------------------
SQL handle: SQL_2ff70e347d63276d
SQL text: SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id,
t1.data, t2.data
FROM t2, t1 WHERE   t1.id = t2.id2 and t1.type=:t
------------------------------------------------------
```

Plan steps are not complete?
=> It is when using format ‘adaptive’

=> It should be adaptive?

SPM (baselines) with adaptive plan - evolution

```
exec :e := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(SQL_HANDLE=>'SQL_2ff70e347d63276d', COMMIT => 'NO');
```

```
Task Information:
 ---------------------------------------------
 Task Name         : TASK_263
 Task Owner        : SCOTT
 Execution Name       : EXEC_263
 Execution Type       : SPM EVOLVE
 Scope             : COMPREHENSIVE
 Status            : COMPLETED
 Started           : 12/03/2015 06:15:01
 Finished          : 12/03/2015 06:15:01
 Last Updated         : 12/03/2015 06:15:01
 Global Time Limit    : 2147483646
 Per-Plan Time Limit  : UNUSED
 Number of Errors     : 0
-------------------------------------------------------------


SUMMARY SECTION
-------------------------------------------------------------
  Number of plans processed  : 1
  Number of findings         : 1
  Number of recommendations  : 1
  Number of errors       : 0
-------------------------------------------------------------
```

```
DETAILS SECTION
-------------------------------------------------------------------------------
 Object ID           : 2
 Test Plan Name      : SQL_PLAN_2zxsf6jyq69vdafb5d283
 Base Plan Name      : SQL_PLAN_2zxsf6jyq69vdcea8bf8c
 SQL Handle          : SQL_2ff70e347d63276d
 Parsing Schema      : SCOTT
 Test Plan Creator   : SCOTT
 SQL Text       : SELECT /*+ GATHER_PLAN_STATISTICS */ t1.id, t1.data,
             t2.data FROM t2, t1 WHERE t1.id = t2.id2 and t1.type=:t

..
Execution Statistics:
-----------------------------
                   Base Plan           Test Plan
                   -----------------   ----------------
 Elapsed Time (s):           .000289           .000025
 CPU Time (s):               .000278           .000011
 Buffer Gets:                48                1
 Optimizer Cost:             139               3
 Disk Reads:                 0                 0
 Direct Writes:              0                 0
 Rows Processed:             0                 0
 Executions:                 10                10
```

SPM (baselines) with adaptive plan - evolution

```
FINDINGS SECTION
-------------------------------------------------------------------------------

Findings (1):
-----------------------------
 1. The plan was verified in 0.08000 seconds. It passed the benefit criterion
    because its verified performance was 25.63122 times better than that of the
    baseline plan.

Recommendation:
-----------------------------
 Consider accepting the plan.
```

Same two plans shown earlier follows in that report, none of them adaptive

Adaptive plans – challenges

- Several ways to see the plan used by a query
  - Autotrace does not show the final plan always
  - dbms_xplan.display_sql_plan_baseline has problems too

- Several execution plans for the same query
  - FULL_PLAN_HASH_VALUE needs to be used
  - But it is different for the same “sql/final plan” if adaptive features were not used

- **When table cardinality changes, plans can change, but only after hard parsing**

Review of Adaptive optimizations features

- Adaptive statistics
  - ✓ Dynamic Statistics
  - ✓ Automatic Reoptimization
  - ✓ SQL Plan Directives
- Adaptive plans
  - ✓ Join Methods
  - **Parallel Distribution Methods**

Adaptive plans - Parallel Distribution Methods

- New hybrid hash distribution method
  - defers decision to use hash or broadcast to execution time

- Collector step in front of parallel coordinator
- Threshold is 2*DOP
- Enabled by default

Adaptive plans - Adaptive bitmap pruning

- This is not documented under query optimizer adaptive features
- New parameter *_optimizer_strans_adaptive_pruning* - allow adaptive pruning of star transformation bitmap trees
- Article by Frank Pachot:
  http://www.slideshare.net/pachot/nl-2014-4adaptivebitmappruning

Questions?

calero@pythian.com

@ncalerouy

http://www.linkedin.com/in/ncalero

References - documentation

Database SQL Tuning Guide – Query optimizer concepts
https://docs.oracle.com/database/121/TGSQL/tgsql_optcncpt.htm#TGSQL192

Optimizer with Oracle Database 12c
http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-optimizer-with-oracledb-12c-1963236.pdf

Understanding Optimizer Statistics with Oracle Database 12c
http://www.oracle.com/technetwork/database/bi-datawarehousing/twp-statistics-concepts-12c-1963871.pdf

Several people blogging about this topics

- Frank Pachot - http://www.slideshare.net/pachot
- Tim Hall - https://oracle-base.com/articles/12c/adaptive-query-optimization-12cr1
- Kerry Osborne - http://kerryosborne.oracle-guy.com/2013/11/12c-adaptive-optimization-part-1/

Database Licensing Information - Oracle Database Editions
https://docs.oracle.com/database/121/DBLIC/editions.htm#DBLIC116

References – views to find information

V$SQL - IS_RESOLVED_ADAPTIVE_PLAN
IS_REOPTIMIZABLE
FULL_PLAN_HASH_VALUE

DBA_SQL_PLAN_DIRECTIVES / DBA_SQL_PLAN_DIR_OBJECTS

DBA_SQL_PLAN_BASELINES - ADAPTIVE

dbms_xplan.display – ADAPTIVE format

V$SQL_PLAN/ DBA_HIST_SQL_PLAN - OTHER_XML
notes of adaptive features used

V$SQL_REOPTIMIZATION_HINTS

References - controlling adaptive features

Initialization parameters for all features:

- OPTIMIZER_ADAPTIVE_FEATURES (Default: TRUE)
  Disables all adaptive features
- OPTIMIZER_ADAPTIVE_REPORTING_ONLY (Default: FALSE)
  Only reports what plans should have been if the feature is enable
- OPTIMIZER_FEATURES_ENABLE – 12.1.0.1 or higher

Parameters per feature

- OPTIMIZER_DYNAMIC_SAMPLING = number => 11 is the new adaptive feature
- _OPTIMIZER_ADAPTIVE_PLANS
- _OPTIMIZER_USE_FEEDBACK

Hints (statement level):

/*+ NO_ADAPTIVE_PLAN */ - new in 12.1.0.2, but statistics feedback works
/*+ ADAPTIVE_PLAN */
/*+ DYNAMIC_SAMPLING */

References - tracing

Global statement trace for already existing and new connections:

```
dbms_sqldiag.dump_trace(p_sql_id=>'0a14b3yhux040', p_child_number=>0,
                        p_component=>'Compiler',  p_file_id=>'trace_0a14b3yhux040');
```

=> This can be used after statement execution

Classic optimizer trace:

```
ALTER SESSION SET EVENTS 'trace[sql_optimizer.*]';
-- same but for Oracle versions older than 11g
ALTER SESSION SET EVENTS='10053 trace name context forever, level 1';
```

SPM Trace at session level:

```
ALTER SESSION SET EVENTS 'trace[RDBMS.SQL_Plan_Management.*]';
```

References - script used - Adaptive plans

```
drop table t1 purge;
create table t1 as
  select rownum id, mod(rownum,100) type,
         object_name data
  from dba_objects;

alter table t1 add constraint t1_pk
      primary key (id);

drop table t2 purge;
create table t2 as
 select rownum id2, mod(rownum,10) type,
        object_name data
 from dba_objects
 where rownum < 100;

alter table t2 add constraint t2_pk
      primary key (id2);

exec dbms_stats.gather_table_stats(USER, 't1');
exec dbms_stats.gather_table_stats(USER ,'t2');
```

```
set autotrace off
col data for a30
SELECT /*+ GATHER_PLAN_STATISTICS */
       t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

select * from table
 (dbms_xplan.display_cursor
  (format=>'adaptive allstats last'));
```

test-join-nl.sql

References - script used - Adaptive plans

```
insert into t2
select rownum+100,
        mod(rownum,10) type,
        object_name data
from dba_objects
where rownum < 1000;

-- no need to update stats for plan to change

set autotrace off
col data for a30
SELECT /*+ GATHER_PLAN_STATISTICS */
       t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

select * from table(dbms_xplan.display_cursor
   (format=>'adaptive allstats last'));

-- Plan didn't changed here, still NL used
-- after parsing sql again, hash_join kicks in
```

```
alter system flush shared_pool;

col data for a30
SELECT /*+ GATHER_PLAN_STATISTICS */
      t1.id, t1.data --, t2.data
FROM   t2, t1
WHERE  t1.id = t2.id2;

select * from table(dbms_xplan.display_cursor
   (format=>'adaptive allstats last'));

SELECT sql_id, child_number, plan_hash_value,
       full_plan_hash_value full_phv,
       is_bind_sensitive BS, is_bind_aware BA,
       IS_REOPTIMIZABLE RE, executions exe,
       IS_RESOLVED_ADAPTIVE_PLAN AP
FROM   v$sql
WHERE sql_id='5bvpk0nnd4323';
```

test-join-hj.sql