
# Getting the most out of your Oracle 12.2+ Optimizer (i.e. The Brain)

Janis Griffin

Senior DBA / Performance Evangelist

# Who Am I

- Senior DBA / Performance Evangelist for SolarWinds
  - Janis.Griffin@solarwinds.com
  - Twitter® - @DoBoutAnything
  - Current – 25+ Years in Oracle®, DB2®, ASE, SQL Server®, MySQL®
  - DBA and Developer
- Specialize in Performance Tuning
- Review Database Performance for Customers and Prospects
- Common Question – How do I tune it?

# Agenda

- 12.2 Optimizer
  - Adaptive Plans
  - Adaptive Statistics
    - Dynamic Statistics / Statistic Feedback review
    - New Sql Plan Directives features explained
    - New Optimizer Statistics Advisor
- SQL Plan Management - What's new?
  - How it coexists with adaptive plans
  - How to control baselines
- Approximate Query Processing
  - What is it
  - When and how to use it

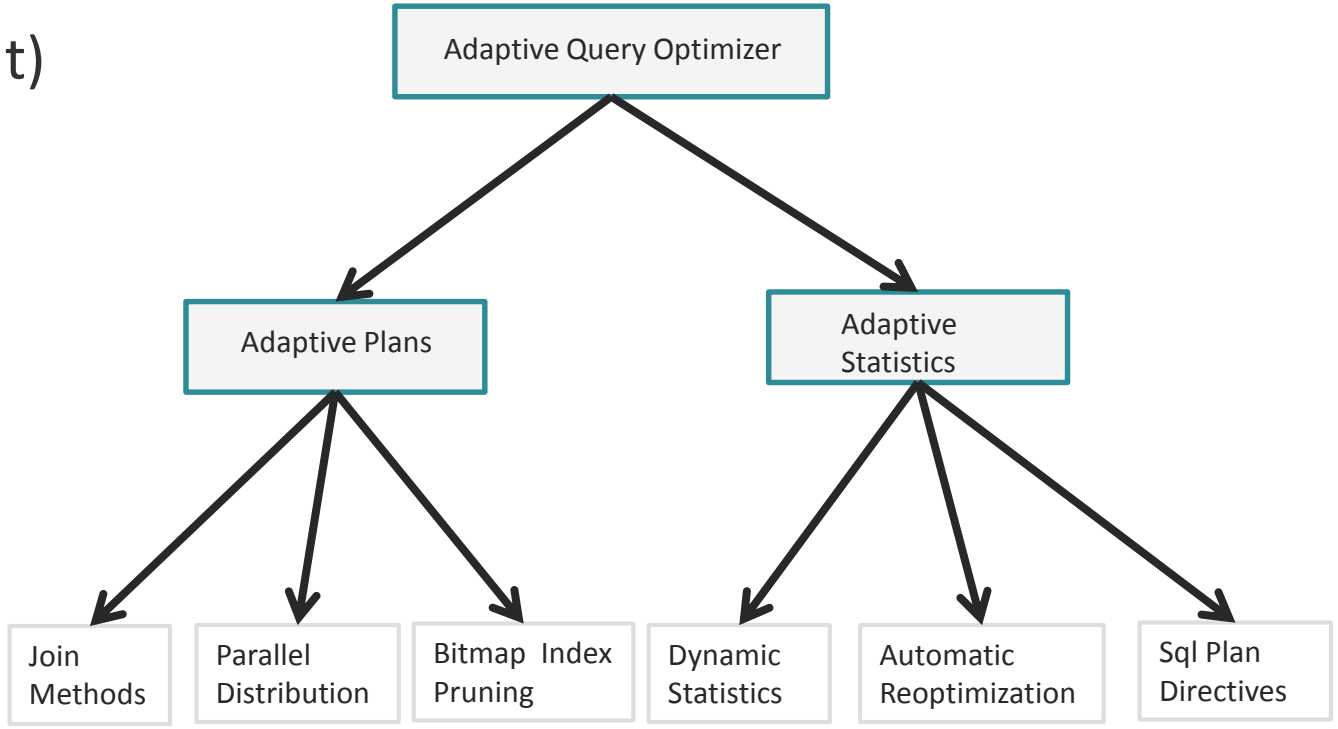
# In The Beginning… Optimizer Overview

solarwinds

- Rule Based Optimizer  (Version < 7.3)
  - Rules based on 17 possible access paths
    - Only one execution plan chosen and only simple rewrites of ‘OR’ to ‘Union ALL’
- Cost Based Optimizer  (Version > = 7.3)
  - Multiple plans generated with estimated cost of IO/CPU
    - Plan with lowest cost chosen
  - Allows for hash joins, histograms, partitioning and parallel queries
  - More complex rewrites and transformations
  - Required statistics gathering  and plans changed
    - 8.1.7, Stored outlines to control plan changes
    - 9.2, Dynamic sampling of statistics
    - 10g, SQL Profiles / Tuning Advisor
      - DBMS_SQLTUNE – Costs $$$
    - Oracle 11,  Adaptive cursor sharing / SQL plan management / SQL patches
    - Oracle 12.1, Adaptive optimizer – could only turn on or off feature

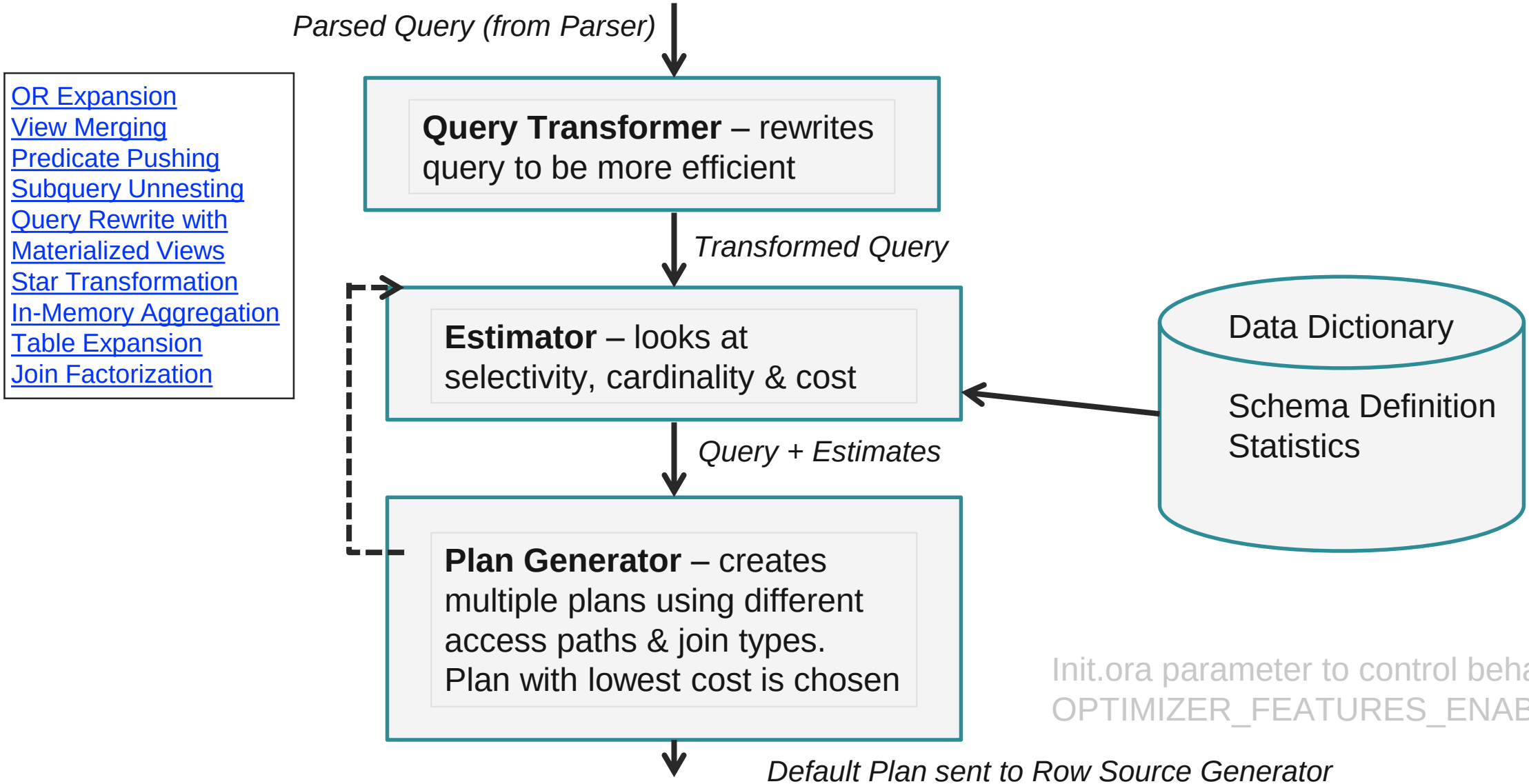
# Adaptive Optimizer 12.2

- Two new parameters
  - Adaptive statistics turned off (default)
    - SQL plan directives
    - Statistics feedback for joins
    - Performance feedback
    - Dynamic sampling for parallel query




| NAME | TYPE | VALUE |
|---|---|---|
| optimizer_adaptive_plans | boolean | TRUE |
| optimizer_adaptive_reporting_only | boolean | FALSE |
| optimizer_adaptive_statistics | boolean | FALSE |
| optimizer_capture_sql_plan_baselines | boolean | FALSE |
| optimizer_dynamic_sampling | integer | 2 |
| optimizer_features_enable | string | 12.2.0.1 |
| optimizer_index_caching | integer | 0 |
| optimizer_index_cost_adj | integer | 100 |
| optimizer_inmemory_aware | boolean | TRUE |
| optimizer_mode | string | ALL_ROWS |
| optimizer_secure_view_merging | boolean | TRUE |
| optimizer_use_invisible_indexes | boolean | FALSE |
| optimizer_use_pending_statistics | boolean | FALSE |
| optimizer_use_sql_plan_baselines | boolean | TRUE |

# How the Optimizer Works



# Execution Plan

solarwinds

- Show the **sequence of operations** performed to run SQL Statement
  - Order of the tables referenced in the statements
  - Access method for each table in the statement
    - INDEX
    - INLIST ITERATOR
    - TABLE ACCESS
    - VIEW
  - Join method in accessing multiple tables
    - HASH JOIN
    - MERGE JOIN
    - NESTED LOOPS
  - Data manipulations
    - CONCATENATION
    - COUNT
    - FILTER
    - SORT
  - Statistic Collectors
    - New in 12C

```
SQL> explain plan for
  2  select order_id, product_name, list_price, unit_price,quantity
  3  from product p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < 1.05;

SQL> select * from table(dbms_xplan.display());

Explained.

Plan hash value: 2408771214
-------------------------------------------------------------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |             |     1 |    55 |     4   (0)|
|   1 |  NESTED LOOPS                 |             |     1 |    55 |     4   (0)|
|   2 |   NESTED LOOPS                |             |     1 |    55 |     4   (0)|
|*  3 |    TABLE ACCESS FULL          | ORDER_ITEMS |     1 |    16 |     3   (0)|
|*  4 |    INDEX UNIQUE SCAN          | PRODUCT_PK  |     1 |       |     0   (0)|
|   5 |   TABLE ACCESS BY INDEX ROWID| PRODUCT     |     1 |    39 |     1   (0)|
-------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   3 - filter("UNIT_PRICE"<1.05)
   4 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
Note
-----
   - this is an adaptive plan
```

# Execution Plan

- Optimizer instruments default plan with statistics collector (SC)
  - SC buffers a portion of rows coming into each sub-plan on initial execution
- Optimizer computes inflection points
  - Determines optimal join type
  - At runtime
- Works only on:
  - Join Methods
    - Nested loops and hash joins
  - Parallel Distribution Method
- No adaptation occurs
  - If initial join is sort merge join

```
SQL> explain plan for
  2  select order_id, product_name, list_price, unit_price,quantity
  3  from product p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < 1.05;

Explained.

SQL> select * from table (dbms_xplan.display( format=> '+adaptive'));

Plan hash value: 2408771214
-------------------------------------------------------------------------------------------
|   Id  | Operation                     | Name        | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |             |     1 |    55 |     4   (0)|
|- *  1 |  HASH JOIN                    |             |     1 |    55 |     4   (0)|
|     2 |   NESTED LOOPS                |             |     1 |    55 |     4   (0)|
|     3 |    NESTED LOOPS               |             |     1 |    55 |     4   (0)|
|-    4 |     STATISTICS COLLECTOR      |             |       |       |            |
|  *  5 |      TABLE ACCESS FULL        | ORDER_ITEMS |     1 |    16 |     3   (0)|
|  *  6 |     INDEX UNIQUE SCAN         | PRODUCT_PK  |     1 |       |     0   (0)|
|     7 |    TABLE ACCESS BY INDEX ROWID| PRODUCT     |     1 |    39 |     1   (0)|
|-    8 |   TABLE ACCESS FULL           | PRODUCT     |     1 |    39 |     1   (0)|
-------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
   5 - filter("UNIT_PRICE"<1.05)
   6 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```

# Optimizer Computes Inflection Points (IPs)

- IPs are statistics where two plan choices are equally good

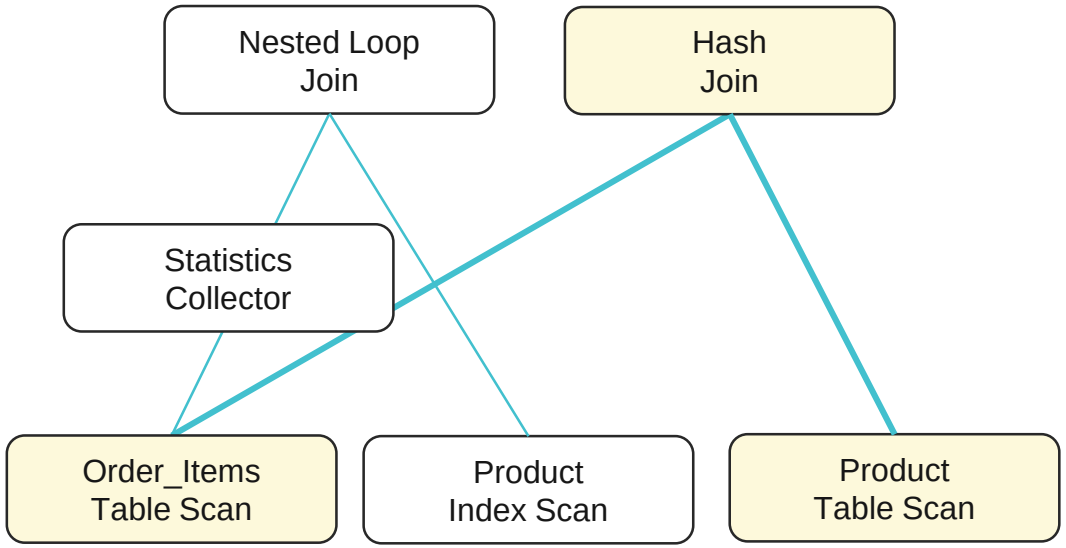
```
SQL> select /*+ gather_plan_statistics  */ order_id,
  2  product_name, list_price, unit_price,quantity
  3  from product p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < 1.05;

  ORDER_ID PRODUCT_NAME               LIST_PRICE UNIT_PRICE QUANTITY
---------- -------------------------- ---------- ---------- --------
      2404 Paper Tablet LY 8 1/2 x 11          1          0       37

SQL> select * from table
  2  (dbms_xplan.display_cursor(format=>'rowstats last adaptive report'));

SQL_ID  38qhzuqntqzq5, child number 0
-------------------------------------
select /*+ gather_plan_statistics  */ order_id, product_name,
list_price, unit_price,quantity from product p, order_items oi where
p.product_id = oi.product_id and unit_price < 1.05

Plan hash value: 2408771214
-----------------------------------------------------------------------------------
|   Id  | Operation                     | Name        | Starts | E-Rows | A-Rows |
-----------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |             |      1 |        |      1 |
|- *  1 |  HASH JOIN                    |             |      1 |      1 |      1 |
|     2 |   NESTED LOOPS                |             |      1 |      1 |      1 |
|     3 |    NESTED LOOPS               |             |      1 |      1 |      1 |
|-    4 |     STATISTICS COLLECTOR      |             |      1 |        |      1 |
|  *  5 |      TABLE ACCESS FULL        | ORDER_ITEMS |      1 |      1 |      1 |
|  *  6 |     INDEX UNIQUE SCAN         | PRODUCT_PK  |      1 |      1 |      1 |
|     7 |    TABLE ACCESS BY INDEX ROWID| PRODUCT     |      1 |      1 |      1 |
|-    8 |   TABLE ACCESS FULL           | PRODUCT     |      0 |      1 |      0 |
-----------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
...cut for brevity...
Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```

```
SQL> alter session set tracefile_identifier='INFLECTION';

SQL> exec dbms_sqldiag.dump_trace(p_sql_id=>'&sql_id', -
> p_child_number=>&child,p_component=>'Compiler',p_file_id=>'');

Enter value for sql_id: 38qhzuqntqzq5
Enter value for child: 0

PL/SQL procedure successfully completed.

$ grep -hE "^DP|^AP"  ora122_ora_*_inflection.trc
DP - distinct placement
AP - adaptive plans
AP: Checking validity for query block SEL$1, sqlid=gdbqk1hmszbxz
AP: Computing costs for inflection point at max value 131070.00
AP: Costing Nested Loops Join for inflection point at card 131070.00
AP: Costing Hash Join for inflection point at card 131070.00
AP: lcost=131104.77, rcost=185.65
AP: Searching for inflection point at value 1.00
AP: Costing Nested Loops Join for inflection point at card 65535.74
AP: Costing Hash Join for inflection point at card 65535.74
...cut for brevity...
AP: Searching for inflection point at value 5.48
AP: Costing Nested Loops Join for inflection point at card 7.48
AP: Costing Hash Join for inflection point at card 7.48
AP: lcost=10.01, rcost=10.03
AP: Searching for inflection point at value 7.48
AP: Costing Nested Loops Join for inflection point at card 8.48
AP: Costing Hash Join for inflection point at card 8.48
AP: lcost=11.01, rcost=10.03
AP: Costing Hash Join for inflection point at card 8.48
DP: Found point of inflection for NLJ vs. HJ: card = 8.48
AP: qesdpResolveSlaveSameAsQC - NOT SLAVE SQL
```

CBO trace or 10053 event (see appendix)

# Optimizer Computes Inflection Points – cont.

- NLJ vs. HJ plans are equally good
  - When cardinality = 8.48




```
SQL> select /*+ gather_plan_statistics  */ order_id,
  2  product_name, list_price, unit_price,quantity
  3  from product p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < 2.1;

25 rows selected.

SQL> select * from table
  2 (dbms_xplan.display_cursor(format=>'rowstats last adaptive report'));

SQL_ID  11bq5qvchns18, child number 0
-------------------------------------
select /*+ gather_plan_statistics  */ order_id, product_name,
list_price, unit_price,quantity from product p, order_items oi where
p.product_id = oi.product_id and unit_price < 2.1

Plan hash value: 158447987

-------------------------------------------------------------------------------------------------
|   Id  | Operation                          | Name          | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT                   |               |      1 |        |     25 |
|  *  1 |  HASH JOIN                         |               |      1 |     26 |     25 |
|-    2 |   NESTED LOOPS                     |               |      1 |     26 |     25 |
|-    3 |    NESTED LOOPS                    |               |      1 |        |     25 |
|-    4 |     STATISTICS COLLECTOR           |               |      1 |        |     25 |
|  *  5 |      TABLE ACCESS FULL             | ORDER_ITEMS   |      1 |     26 |     25 |
|- *  6 |     INDEX UNIQUE SCAN              | PRODUCT_PK    |      0 |        |      0 |
|-    7 |    TABLE ACCESS BY INDEX ROWID     | PRODUCT       |      0 |      1 |      0 |
|     8 |   TABLE ACCESS FULL                | PRODUCT       |      1 |    288 |    288 |
-------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
   5 - filter("UNIT_PRICE"<2.1)
   6 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
Note
-----
   - this is an adaptive plan (rows marked '-' are inactive)
```

# How To Identify Adaptive Plans

solarwinds

- New columns in V$SQL:
  - IS_RESOLVED_ADAPTIVE_PLAN
    - If 'Y', the plan was adapted & is the final plan
    - If 'N', the plan is adaptive
      - But final plan has not been selected
    - If NULL, the plan is non-adaptive
  - IS_REOPTIMIZABLE is for next execution
    - Y - the next execution will trigger a reoptimization
    - R – has reoptimization info
      - But won't trigger due to reporting mode
    - N -the child cursor has no reoptimization info

alter (system / session) set optimizer_adaptive_reporting_only=TRUE;

```
SQL> select sql_id,child_number,substr(sql_text,1,30) sql_text,
  2  IS_RESOLVED_ADAPTIVE_PLAN, IS_REOPTIMIZABLE
  3  from v$sql
  4  where plan_hash_value = 1581076404;

SQL_ID        CHILD_NUMBER SQL_TEXT                      I I
------------- ------------ ------------------------------ - -
fn6x77x1h7s6q            0  explain plan for   select  or Y N
fn6x77x1h7s6q            1  explain plan for   select  or Y N
fn6x77x1h7s6q            2  explain plan for   select  or Y N
```

```
SQL> explain plan for
  2  select  order_id, product_name, list_price, UNIT_PRICE,QUANTITY
  3  from products p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < :b1;

Explained.

SQL> select * from table (dbms_xplan.display( format=> '+adaptive'));

Plan hash value: 1581076404
-------------------------------------------------------------------------------
|   Id  | Operation | Name                    | Rows  | Bytes | Cost (%CPU)
-------------------------------------------------------------------------------
|     0 | SELECT STATEMENT                |   |    65 |  5200 |    56   (0)
|     1 |  NESTED LOOPS OUTER             |   |    65 |  5200 |    56   (0)
|  *  2 |   HASH JOIN                     |   |    48 |  1920 |     8   (0)
|-    3 |    NESTED LOOPS                 |   |    48 |  1920 |     8   (0)
|-    4 |     STATISTICS COLLECTOR        |   |       |       |
|  *  5 |      TABLE ACCESS FULL          | ORDER_ITEMS           |    48
|-    6 |     TABLE ACCESS BY INDEX ROWID| PRODUCT_INFORMATION    |     1
|- *  7 |      INDEX UNIQUE SCAN          | PRODUCT_INFORMATION_PK |
|     8 |    TABLE ACCESS FULL            | PRODUCT_INFORMATION    |   288
|     9 |   TABLE ACCESS BY INDEX ROWID   | PRODUCT_DESCRIPTIONS   |     1
|  * 10 |    INDEX UNIQUE SCAN            | PRD_DESC_PK            |     1
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("I"."PRODUCT_ID"="OI"."PRODUCT_ID")
   5 - filter("UNIT_PRICE"<TO_NUMBER(:B1))
   7 - access("I"."PRODUCT_ID"="OI"."PRODUCT_ID")
  10 - access("D"."PRODUCT_ID"(+)="I"."PRODUCT_ID" AND
              "D"."LANGUAGE_ID"(+)=SYS_CONTEXT('USERENV','LANG'))

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

# Why Plans Changes

- Execution plans can change as underlying inputs to optimizer change
  - Same Sql – Different Schemas
    - Different  table sizes / statistics / indexes
  - Same Sql – Different Costs
    - Data volume & Statistic Changes over time
    - Bind variable types and values
    - Initialization parameters (set globally or session level)
    - Adaptive Cursor Sharing – 11G
      - V$SQL - IS_BIND_SENSITIVE: optimizer peeked –plan may change
      - V$SQL - IS_BIND_AWARE:  'Y' after query has been marked bind sensitive
    - Adaptive Plans / Statistics – 12C
  - V$SQL_SHARED_CURSOR
    - Can give clues to why plan changed
    - 70 columns showing mismatches /differences
    - Hard to view
    - Script, 'shared_proc.sql' in appendix

# V$SQL_SHARED_CURSOR

solarwinds

- Shared_proc.sql script in appendix

```
SQL_TEXT      =  explain plan for   select  order_id, product_name, list_price, UNIT_PRICE,
p, order_items oi   where p.product_id = oi.product_id   and unit_price < :b1
SQL_ID        = fn6x77x1h7s6q
ADDRESS       = 000000006BF7F3E0
CHILD_ADDRESS = 00000000616111C8
CHILD_NUMBER  = 0
REASON        = <ChildNode><ChildNumber>0</ChildNumber><ID>20</ID><reason>Explain Plan cur
_cursor>50</ctxflg_cursor><Literal_Replacement_Enabled>1</Literal_Replacement_Enabled></Chi
---------------------------------------------------
SQL_TEXT            =  explain plan for   select  order_id, product_name, list_price, UNIT_
p, order_items oi   where p.product_id = oi.product_id   and unit_price < :b1
SQL_ID              = fn6x77x1h7s6q
ADDRESS             = 000000006BF7F3E0
CHILD_ADDRESS       = 00000000A9C29D48
CHILD_NUMBER        = 1
EXPLAIN_PLAN_CURSOR = Y
REASON              = <ChildNode><ChildNumber>1</ChildNumber><ID>20</ID><reason>Explain Pla
ctxflg_cursor>50</ctxflg_cursor><Literal_Replacement_Enabled>1</Literal_Replacement_Enable
ldNode>
---------------------------------------------------
SQL_TEXT            =  explain plan for   select  order_id, product_name, list_price, UNIT_
p, order_items oi   where p.product_id = oi.product_id   and unit_price < :b1
SQL_ID              = fn6x77x1h7s6q
ADDRESS             = 000000006BF7F3E0
CHILD_ADDRESS       = 000000009CA34290
CHILD_NUMBER        = 2
EXPLAIN_PLAN_CURSOR = Y
REASON= <ChildNode><ChildNumber>2</ChildNumber><ID>3</ID><reason>Optimizer mismatch(12)</re
ptive_reporting_only> false                 true       </optimizer_adaptive_reporting_only>
---------------------------------------------------
```

Explain Plans are blind to bind variables

Adaptive Reporting Only Mode

# Get Real Execution Plan (reporting_only)

```
SQL> exec :b1 := 10.25;
SQL> select /*+ gather_plan_statistics */ order_id,
  2  product_name, list_price, unit_price,quantity
  3  from product p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < :b1;

63 rows selected.

SQL> select * from table
  2 (dbms_xplan.display_cursor('bcx68bhgfjrtj','0',
  3 format=>'rowstats last adaptive report'));

SQL_ID  bcx68bhgfjrtj, child number 0
-------------------------------------
select /*+ gather_plan_statistics  */ order_id, product_name,
list_price, unit_price,quantity from product p, order_items oi where
p.product_id = oi.product_id and unit_price < :b1

Plan hash value: 158447987
-------------------------------------------------------------------------------------
|   Id  | Operation                      | Name        | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT               |             |      1 |        |     63 |
|  *  1 |  HASH JOIN                     |             |      1 |     63 |     63 |
|-    2 |   NESTED LOOPS                 |             |      1 |     63 |     63 |
|-    3 |    NESTED LOOPS                |             |      1 |        |     63 |
|-    4 |     STATISTICS COLLECTOR       |             |      1 |        |     63 |
|  *  5 |      TABLE ACCESS FULL         | ORDER_ITEMS |      1 |     63 |     63 |
|- *  6 |     INDEX UNIQUE SCAN          | PRODUCT_PK  |      0 |        |      0 |
|-    7 |    TABLE ACCESS BY INDEX ROWID| PRODUCT     |      0 |      1 |      0 |
|     8 |   TABLE ACCESS FULL            | PRODUCT     |      1 |    288 |    288 |
-------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
   5 - filter("UNIT_PRICE"<:B1)
   6 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")

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

```
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: 158447987
-------------------------------------------------------
| Id  | Operation           | Name        | Rows  |
-------------------------------------------------------
|   0 | SELECT STATEMENT    |             |       |
|*  1 |  HASH JOIN          |             |    63 |
|*  2 |   TABLE ACCESS FULL| ORDER_ITEMS |    63 |
|   3 |   TABLE ACCESS FULL| PRODUCT     |   288 |
-------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("P"."PRODUCT_ID"="OI"."PRODUCT_ID")
   2 - filter("UNIT_PRICE"<:B1)

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

```
SQL> select sql_id,child_number,substr(sql_text,1,30) sql_text,
  2  IS_RESOLVED_ADAPTIVE_PLAN, IS_REOPTIMIZABLE,
  3  IS_BIND_AWARE,IS_BIND_SENSITIVE
  4  from v$sql
  5  where sql_text like 'select %gather%';

SQL_ID        CHILD_NUMBER SQL_TEXT                       I I I I
------------- ------------ ------------------------------ - - - -
bcx68bhgfjrtj            0 select /*+ gather_plan_statist Y N N Y
```

# Get Real Execution Plan

- Optimizer Mismatch

```
SQL> alter session set optimizer_adaptive_reporting_only=FALSE;

Session altered.

SQL> select /*+ gather_plan_statistics  */ order_id,
  2  product_name, list_price, unit_price,quantity
  3  from product p, order_items oi
  4  where p.product_id = oi.product_id
  5  and unit_price < :b1;

63 rows selected.

SQL> select sql_id,child_number,substr(sql_text,1,30) sql_text,
  2  IS_RESOLVED_ADAPTIVE_PLAN, IS_REOPTIMIZABLE,
  3  IS_BIND_AWARE,IS_BIND_SENSITIVE
  4  from v$sql
  5  where sql_text like 'select %gather%';

SQL_ID        CHILD_NUMBER SQL_TEXT    I I I I
------------- ------------ ------------------------------ - - - -
bcx68bhgfjrtj            0 select /*+ gather_plan_statist Y N N Y
bcx68bhgfjrtj            1 select /*+ gather_plan_statist Y N N Y
```

```
SQL> exec :b1 := 1000000.5

SQL> @sq1
665 rows selected.

SQL_ID        CHILD_NUMBER SQL_TEXT                       I I I I
------------- ------------ ------------------------------ - - - -
bcx68bhgfjrtj            0 select /*+ gather_plan_statist Y N N Y
bcx68bhgfjrtj            1 select /*+ gather_plan_statist Y N N Y
bcx68bhgfjrtj            2 select /*+ gather_plan_statist Y N Y Y

SQL> select * from table(
  2  dbms_xplan.display_cursor('bcx68bhgfjrtj','2'));

SQL_ID  bcx68bhgfjrtj, child number 2
-------------------------------------
select /*+ gather_plan_statistics  */ order_id, product_name,
list_price, unit_price,quantity from product p, order_items oi where
p.product_id = oi.product_id and unit_price < :b1

Plan hash value: 618102895

-------------------------------------------------------------------------------
| Id  | Operation          | Name        | Rows  | Bytes | Cost (%CPU)|
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |             |       |       |    10 (100)|
|*  1 |  HASH JOIN         |             |   665 | 36575 |    10   (0)|
|   2 |   TABLE ACCESS FULL| PRODUCT     |   288 | 11232 |     7   (0)|
|*  3 |   TABLE ACCESS FULL| ORDER_ITEMS |   665 | 10640 |     3   (0)|
-------------------------------------------------------------------------------
```

# Adaptive Cursor Sharing

solarwinds

- V$SQL_CS_SELECTIVITY (BIND AWARE)

```
SQL> select SQL_ID,CHILD_NUMBER,PREDICATE,RANGE_ID,LOW,HIGH
  2  from  v$sql_cs_selectivity
  3  where sql_id = 'bcx68bhgfjrtj' order by 2;

SQL_ID        CHILD_NUMBER PREDICATE                 RANGE_ID LOW         HIGH
------------- ------------ -------------------- ---------- ---------- ----------
bcx68bhgfjrtj            2 <B1                           0 0.900000   1.100000
bcx68bhgfjrtj            3 <B1                           0 0.732857   1.100000
bcx68bhgfjrtj            4 <B1                           0 0.001999   0.002444
```

:b1 = 1000000.5
:b1 = 150.0
:b1 = 1.05

- V$SQL_CS_HISTOGRAM
  - Summary of monitoring
  - Three buckets (S/M/L)
- V$SQL_CS_STATISTICS
  - Show rows processed
  - Empty in 12.2?

```
SQL> select SQL_ID,CHILD_NUMBER,BUCKET_ID,COUNT from v$sql_cs_histogram
  2  where SQL_ID = 'b9nbhsbx8tqz5'
  3  order by sql_id, child_number;

SQL_ID                 CHILD_NUMBER            BUCKET_ID                COUNT
------------- -------------------- -------------------- --------------------
b9nbhsbx8tqz5                    2                    1                    0
b9nbhsbx8tqz5                    2                    0                    2
b9nbhsbx8tqz5                    2                    2                    0
b9nbhsbx8tqz5                    4                    0                    8
b9nbhsbx8tqz5                    4                    1                    0
b9nbhsbx8tqz5                    4                    2                    0
b9nbhsbx8tqz5                    5                    1                    0
b9nbhsbx8tqz5                    5                    0                 1000
b9nbhsbx8tqz5                    5                    2                    0
```

# Parallel Distribution Method

solarwinds

- Optimizer must distribute data across all parallel processes
  - Parallel queries are useful for sorts, aggregation & join operations
- Optimizer chooses between broadcast or hybrid hash (NEW in 12c)
  - Chosen method  depends on number of rows and Degree of Parallelism (DOP)
    - Hybrid hash, if rows > than threshold
    - Broadcast, if rows < than threshold
    - Threshold is defined as 2 X DOP
- Optimizer decides final data distribution method during each execution time
  - Different from adaptive joins which are limited to first execution only
  - Statistic collectors are inserted in front of the parallel server processes
    - On producer side of the operation
- Hybrid hash distribution can help with data skew
  - Potential performance problem if few parallel processes distribute many rows

# Parallel Distribution - Hybrid Hash Example

solarwinds

- Alter session set optimizer_adaptive_statistics = TRUE;

```
SQL> exec  :b1 := 79.71
SQL> exec :b2 :=9

SQL> SELECT /*+ GATHER_PLAN_STATISTICS  PARALLEL(10) */ i_name
  2  from order_line ol, item i
  3  where ol_amount = :b1 and ol_quantity > :b2
  4  and ol_i_id = i_id;

I_NAME
------------------------
5NinakyltO51jza8zlQ
...
21 rows selected.
```

- Other parameters
  - parallel_degree_policy (Manual)
    - Can be Adaptive or Auto
  - parallel_adaptive_multi_user
  - parallel_degree_limit
  - parallel_min_time_threshold
- v$pq_tqstat
  - Valid only in current session
  - Shows message traffic
    - At table queue level

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.display_cursor(format => 'allstats last adaptive'));

SQL_ID  7861ww3vwnt7a, child number 2
-------------------------------------
 SELECT /*+ GATHER_PLAN_STATISTICS  PARALLEL(10) */ i_name from
order_line ol, item i where ol_amount = :b1 and ol_quantity > :b2 and ol_i_id = i_id

Plan hash value: 1339474084
-----------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name     | Starts | E-Rows | A-Rows |   A-Time    | Buffers |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |          |      1 |        |     21 |00:00:22.10 |      38 |
|   1 |  PX COORDINATOR               |          |      1 |        |     21 |00:00:22.10 |      38 |
|   2 |   PX SEND QC (RANDOM)         | :TQ10002 |      0 |  30516 |      0 |00:00:00.01 |       0 |
|*  3 |    HASH JOIN BUFFERED         |          |      0 |  30516 |      0 |00:00:00.01 |       0 |
|   4 |     JOIN FILTER CREATE        | :BF0000  |      0 |  30516 |      0 |00:00:00.01 |       0 |
|   5 |      PX RECEIVE               |          |      0 |  30516 |      0 |00:00:00.01 |       0 |
|   6 |       PX SEND HYBRID HASH     | :TQ10000 |      0 |  30516 |      0 |00:00:00.01 |       0 |
|   7 |        STATISTICS COLLECTOR   |          |      0 |        |      0 |00:00:00.01 |       0 |
|   8 |         PX BLOCK ITERATOR     |          |      0 |  30516 |      0 |00:00:00.01 |       0 |
|*  9 |          INDEX FAST FULL SCAN | IORDL    |      0 |  30516 |      0 |00:00:00.01 |       0 |
|  10 |     PX RECEIVE                |          |      0 |   101K |      0 |00:00:00.01 |       0 |
|  11 |      PX SEND HYBRID HASH      | :TQ10001 |      0 |   101K |      0 |00:00:00.01 |       0 |
|  12 |       JOIN FILTER USE         | :BF0000  |      0 |   101K |      0 |00:00:00.01 |       0 |
|  13 |        PX BLOCK ITERATOR      |          |      0 |   101K |      0 |00:00:00.01 |       0 |
|* 14 |         TABLE ACCESS FULL     | ITEM     |      0 |   101K |      0 |00:00:00.01 |       0 |
-----------------------------------------------------------------------------------------------------
  ...cut for brevity...
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - Degree of Parallelism is 10 because of hint
   - statistics feedback used for this statement
   - performance feedback used for this statement
```

# Bitmap Index Pruning (Star Schemas)

solarwinds

- Adaptive plans will prune out insufficient bitmap indexes
  - STAR_TRANSFORMATION_ENABLED parameter must be enabled (TRUE)
    - Default is FALSE
  - Hidden parameter: _optimizer_strans_adaptive_pruning
    - see appendix for script to get  all 12.2 hidden optimizer parameters
- If optimizer generates a star transformation plan
  - Needs to decide which bitmap indexes to use
    - If too many indexes, some may not reduce the numbers of rows returned
      - Can cause unnecessary cost and overhead

```
SELECT  /*+ GATHER_PLAN_STATISTICS */ o.carrier, uc.description AS carrier_name ,
ao.description AS origin_airport,co.Description AS origin_city ,o.fl_date,o.fl_num,
o.tail_num ,ad.description AS destination_airport ,cd.Description AS destination_city,
w.Description Day_of_Week
FROM t_ontime o INNER JOIN L_UNIQUE_CARRIERS uc ON uc.Code = o.UNIQUE_CARRIER
         INNER JOIN L_AIRPORT_ID ao ON ao.Code = o.ORIGIN_AIRPORT_ID
         INNER JOIN L_AIRPORT_ID ad ON ad.Code = o.DEST_AIRPORT_ID
         INNER JOIN L_CITY_MARKET_ID co ON co.Code = o.ORIGIN_CITY_MARKET_ID
         INNER JOIN L_CITY_MARKET_ID cd ON cd.Code = o.DEST_CITY_MARKET_ID
         INNER JOIN L_WEEKDAYS w ON w.Code = o.DAY_OF_WEEK
WHERE to_date(fl_date,'YYYY-MM-DD') BETWEEN to_date('08/01/15','mm/dd/yy')
AND to_date('12/01/15','mm/dd/yy') AND co.Description = 'New York City, NY (Metropolitan Area)'
AND cd.Description = 'Denver, CO' AND ao.Description = 'New York, NY: LaGuardia' AND w.Description = 'Sunday'
```

US DOT - On-time Performance

# Bitmap Index Pruning

solarwinds

```
Plan hash value: 2123677140
-----------------------------------------------------------------------------------------------------------------------
|    Id   | Operation                                    | Name                 | Starts | E-Rows | A-Rows |
-----------------------------------------------------------------------------------------------------------------------
|      0 | SELECT STATEMENT                             |                      |      1 |        |    446 |
|   *  1 |  FILTER                                      |                      |      1 |        |    446 |
|   *  2 |   HASH JOIN                                  |                      |      1 |     47 |    446 |
|   *  3 |    HASH JOIN                                 |                      |      1 |     47 |    446 |
|   *  4 |     HASH JOIN                                |                      |      1 |     47 |    446 |
|   *  5 |      TABLE ACCESS FULL                       | L_WEEKDAYS           |      1 |      1 |      1 |
|   *  6 |      HASH JOIN                               |                      |      1 |     47 |    446 |
|   *  7 |       TABLE ACCESS FULL                      | L_CITY_MARKET_ID     |      1 |      1 |      1 |
|   *  8 |       HASH JOIN                              |                      |      1 |     47 |    446 |
|   *  9 |        TABLE ACCESS FULL                     | L_CITY_MARKET_ID     |      1 |      1 |      1 |
|   * 10 |        HASH JOIN                             |                      |      1 |     47 |  12300 |
|   * 11 |         TABLE ACCESS FULL                    | L_AIRPORT_ID         |      1 |      1 |      1 |
|   * 12 |         TABLE ACCESS BY INDEX ROWID BATCHED  | T_ONTIME             |      1 |     47 |  12300 |
|     13 |          BITMAP CONVERSION TO ROWIDS         |                      |      1 |        |  32458 |
|     14 |           BITMAP AND                         |                      |      1 |        |      2 |
|     15 |            BITMAP MERGE                      |                      |      1 |        |      8 |
|     16 |             BITMAP KEY ITERATION             |                      |      1 |        |    210 |
|   * 17 |              TABLE ACCESS FULL               | L_CITY_MARKET_ID     |      1 |      1 |      1 |
|   * 18 |              BITMAP INDEX RANGE SCAN         | O_D_CITY_MARKET_BMX  |      1 |        |    210 |
|-    19 |            STATISTICS COLLECTOR              |                      |      1 |        |     17 |
|     20 |            BITMAP MERGE                      |                      |      1 |        |     17 |
|     21 |             BITMAP KEY ITERATION             |                      |      1 |        |    153 |
|   * 22 |              TABLE ACCESS FULL               | L_WEEKDAYS           |      1 |      1 |      1 |
|   * 23 |              BITMAP INDEX RANGE SCAN         | DAY_OF_WEEK_BMX      |      1 |        |    153 |
|-    24 |            STATISTICS COLLECTOR              |                      |      1 |        |     32 |
|-    25 |            BITMAP MERGE                      |                      |      1 |        |     32 |
|-    26 |             BITMAP KEY ITERATION             |                      |      1 |        |    281 |
|- * 27 |              TABLE ACCESS FULL               | L_AIRPORT_ID         |      1 |      1 |      1 |
|- * 28 |              BITMAP INDEX RANGE SCAN         | O_AIRPORT_BMX        |      1 |        |    281 |
|     29 |     TABLE ACCESS FULL                        | L_UNIQUE_CARRIERS    |      1 |   1620 |   1620 |
|     30 |    TABLE ACCESS FULL                         | L_AIRPORT_ID         |      1 |   6438 |   6438 |
-----------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter(TO_DATE('12/01/15','mm/dd/yy')>=TO_DATE('08/01/15','mm/dd/yy'))
...cut for brevity...
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - star transformation used for this statement
   - this is an adaptive plan (rows marked '-' are inactive)
```

# Adaptive Statistics: Quality of Plan = Quality of Statistics

- Optimizer can re-optimize a query several times
  - Learning more info and further improving the plan
- Dynamic statistics
- Automatic reoptimization
  - Statistics feedback
  - Performance feedback
- Sql plan directives

What's wrong with these pictures?

# Dynamic Statistics

solarwinds

- Augment missing or insufficient base table statistics
  - Table / index block counts
  - Table / join cardinalities (estimated number of rows)
  - Join column statistics
  - GROUP BY statistics
- Are gathered during the parse stage
  - Uses recursive SQL to scan a random sample of table blocks
  - Statistics gathered are not as high a quality as DBMS_STATS
    - Due to sampling
- Controlled by dynamic sampling init.ora parameter
  - OPTIMIZER_DYNAMIC_SAMPLING
  - New in 12.1 - level 11
    - Automatically controls the creation of dynamic statistics

```
alter session set OPTIMIZER_DYNAMIC_SAMPLING = 11;
```

- Results are stored to minimize performance impact
  - 12.1 - In Server Result Cache
  - 12.2 - As SQL Plan Directive

# Dynamic Statistics Example

```
SQL> explain plan for
  2  select O_ID,O_C_ID,O_CARRIER_ID,O_OL_CNT,O_ENTRY_D
  3  from orders
  4  where O_W_ID=2
  5  and O_D_ID=3;

Explained.

SQL> select * from table (dbms_xplan.display( format=> '+adaptive'));

Plan hash value: 1275100350
-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |   304K|  8627K|   8553   (1)| 00:00:
|*  1 |  TABLE ACCESS FULL| ORDERS    |   304K|  8627K|   8553   (1)| 00:00:
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("O_D_ID"=3 AND "O_W_ID"=2)
```

```
SQL> alter session set optimizer_dynamic_sampling=11;

Session altered.

SQL> explain plan for
  2  select O_ID,O_C_ID,O_CARRIER_ID,O_OL_CNT,O_ENTRY_D
  3  from orders
  4  where O_W_ID=2
  5  and O_D_ID=3;

Explained.

SQL> select * from table (dbms_xplan.display( format=> '+adaptive'));

Plan hash value: 1275100350
-------------------------------------------------------------------------------
| Id  | Operation          | Name     | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |          |   159K|  4530K|   8552   (1)| 00:00:01
|*  1 |  TABLE ACCESS FULL| ORDERS    |   188K|  5337K|   8552   (1)| 00:00:01
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - filter("O_D_ID"=3 AND "O_W_ID"=2)

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

Estimates 2X off!

Notes:
Parse time takes longer.
Results are persisted & used elsewhere

# Statistics Feedback

solarwinds

- Optimizer enables monitoring for statistics feedback when
  - Missing statistics, inaccurate statistics, or complex predicates
    - Such as multiple conjunctive or disjunctive filter predicates on a table
- After 1st execution, estimates are compared with actual rows
  - If they differ significantly, optimizer stores correct estimates for future use
    - Stored as OPT_ESTIMATE hints in V$SQL_REOPTIMIZATION_HINTS
    - Can create a SQL PLAN DIRECTIVE for other SQL statements
  - If they differ, the cursor is marked IS_REOPTIMIZABLE
    - IS_REOPTIMIZABLE column in V$SQL is updated to 'Y'
    - Cursor will not be used again
  - After 1st execution, optimizer disables statistics collectors
- Next execution will incur a hard parse
  - Optimizer uses the statistics found during 1st execution to determine better plan

# Statistics Feedback Example

```
SQL_ID  1dch7djzpyg38, child number 0
-------------------------------------
select /*+ GATHER_PLAN_STATISTICS */ c.cust_first_name,c.cust_last_name,o.order_id,
o.order_status,o.order_total,i.line_item_id, p.product_name, i.unit_price,i.quantity
from customers c, orders o, order_items i, product p where c.customer_id = o.customer_id
and o.order_id = i.order_id and i.product_id = p.product_id and
(cust_first_name like '%i%' or c.cust_last_name like '%gri%')
and p.product_id in (select p.product_id from inventories i, product p
where i.product_id = p.product_id and i.quantity_on_hand> 10 and product_name like '%a%')

Plan hash value: 361536262
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |      1 |        |    733 |00:00:00.01 |     156 |
|*  1 |  HASH JOIN                   |             |      1 |     81 |    733 |00:00:00.01 |     156 |
|*  2 |   TABLE ACCESS FULL          | CUSTOMERS   |      1 |     31 |    118 |00:00:00.01 |      16 |
|*  3 |   HASH JOIN                  |             |      1 |    123 |    936 |00:00:00.01 |     140 |
|*  4 |    HASH JOIN                 |             |      1 |    123 |    936 |00:00:00.01 |      79 |
|*  5 |     HASH JOIN RIGHT SEMI     |             |      1 |     14 |     70 |00:00:00.01 |      63 |
|   6 |      VIEW                    | VW_NSO_1    |      1 |     14 |     70 |00:00:00.01 |      35 |
|*  7 |       HASH JOIN SEMI         |             |      1 |     14 |     70 |00:00:00.01 |      35 |
|*  8 |        TABLE ACCESS FULL     | PRODUCT     |      1 |     14 |    101 |00:00:00.01 |      28 |
|*  9 |        TABLE ACCESS FULL     | INVENTORIES |      1 |   1080 |   1069 |00:00:00.01 |       7 |
|  10 |      TABLE ACCESS FULL       | PRODUCT     |      1 |    288 |    288 |00:00:00.01 |      28 |
|  11 |     TABLE ACCESS FULL        | ORDER_ITEMS |      1 |   2526 |   2526 |00:00:00.01 |      16 |
|* 12 |    TABLE ACCESS FULL         | ORDERS      |      1 |    840 |    840 |00:00:00.01 |      61 |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
   2 - filter(("CUST_FIRST_NAME" LIKE '%i%' OR "C"."CUST_LAST_NAME" LIKE '%gri%'))
   3 - access("O"."ORDER_ID"="I"."ORDER_ID")
   4 - access("I"."PRODUCT_ID"="P"."PRODUCT_ID")
   5 - access("P"."PRODUCT_ID"="PRODUCT_ID")
   7 - access("I"."PRODUCT_ID"="P"."PRODUCT_ID")
   8 - filter(("PRODUCT_NAME" LIKE U'%a%' AND "PRODUCT_NAME" IS NOT NULL))
   9 - filter("I"."QUANTITY_ON_HAND">10)
  12 - filter("O"."CUSTOMER_ID">0)

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

```
SQL_ID  1dch7djzpyg38, child number 1
-------------------------------------
select /*+ GATHER_PLAN_STATISTICS */ c.cust_first_name,c.cust_last_name,o.order_id,
o.order_status,o.order_total,i.line_item_id, p.product_name, i.unit_price,i.quantity
from customers c, orders o, order_items i, product p where c.customer_id = o.customer_id
and o.order_id = i.order_id and i.product_id = p.product_id and
(cust_first_name like '%i%' or c.cust_last_name like '%gri%')
and p.product_id in (select p.product_id from inventories i, product p
where i.product_id = p.product_id and i.quantity_on_hand> 10 and product_name like '%a%')

Plan hash value: 2503326313
-----------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name        | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-----------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |             |      1 |        |    733 |00:00:00.02 |     205 |
|*  1 |  HASH JOIN                   |             |      1 |    733 |    733 |00:00:00.02 |     205 |
|*  2 |   TABLE ACCESS FULL          | CUSTOMERS   |      1 |    118 |    118 |00:00:00.01 |      16 |
|*  3 |   HASH JOIN                  |             |      1 |    936 |    936 |00:00:00.01 |     144 |
|*  4 |    TABLE ACCESS FULL         | ORDERS      |      1 |    840 |    840 |00:00:00.01 |      16 |
|*  5 |    HASH JOIN                 |             |      1 |    936 |    936 |00:00:00.01 |     128 |
|*  6 |     HASH JOIN RIGHT SEMI     |             |      1 |     70 |     70 |00:00:00.01 |      63 |
|   7 |      VIEW                    | VW_NSO_1    |      1 |     70 |     70 |00:00:00.01 |      35 |
|*  8 |       HASH JOIN SEMI         |             |      1 |     70 |     70 |00:00:00.01 |      35 |
|*  9 |        TABLE ACCESS FULL     | PRODUCT     |      1 |    101 |    101 |00:00:00.01 |      28 |
|* 10 |        TABLE ACCESS FULL     | INVENTORIES |      1 |   1080 |   1069 |00:00:00.01 |       7 |
|  11 |      TABLE ACCESS FULL       | PRODUCT     |      1 |    288 |    288 |00:00:00.01 |      28 |
|  12 |     TABLE ACCESS FULL        | ORDER_ITEMS |      1 |   2526 |   2526 |00:00:00.01 |      65 |
-----------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
   2 - filter(("CUST_FIRST_NAME" LIKE '%i%' OR "C"."CUST_LAST_NAME" LIKE '%gri%'))
   3 - access("O"."ORDER_ID"="I"."ORDER_ID")
   4 - filter("O"."CUSTOMER_ID">0)
   5 - access("I"."PRODUCT_ID"="P"."PRODUCT_ID")
   6 - access("P"."PRODUCT_ID"="PRODUCT_ID")
   8 - access("I"."PRODUCT_ID"="P"."PRODUCT_ID")
   9 - filter(("PRODUCT_NAME" LIKE U'%a%' AND "PRODUCT_NAME" IS NOT NULL))
  10 - filter("I"."QUANTITY_ON_HAND">10)

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

# Statistics Feedback Example – Cont.

```
SQL> select SQL_ID,CHILD_NUMBER,HINT_ID,HINT_TEXT
  2  from v$sql_reoptimization_hints where sql_id = '1dch7djzpyg38';

SQL_ID             CHILD    HINT_ID HINT_TEXT
------------- ---------- ---------- ----------------------------------------------------
1dch7djzpyg38          0          1 OPT_ESTIMATE (@"SEL$5DA710D3" JOIN ("C"@"SEL$1" "O
                                     "@"SEL$1" "I"@"SEL$1" "VW_NSO_1"@"SEL$5DA710D3" "P
                                     "@"SEL$1") ROWS=733.000000 )

1dch7djzpyg38          0          2 OPT_ESTIMATE (@"SEL$5DA710D3" JOIN ("O"@"SEL$1" "I
                                     "@"SEL$1" "VW_NSO_1"@"SEL$5DA710D3" "P"@"SEL$1") R
                                     OWS=936.000000 )

1dch7djzpyg38          0          3 OPT_ESTIMATE (@"SEL$5DA710D3" JOIN ("I"@"SEL$1" "V
                                     W_NSO_1"@"SEL$5DA710D3" "P"@"SEL$1") ROWS=936.0000
                                     00 )

1dch7djzpyg38          0          4 OPT_ESTIMATE (@"SEL$5DA710D3" JOIN ("VW_NSO_1"@"SE
                                     L$5DA710D3" "P"@"SEL$1") ROWS=70.000000 )

1dch7djzpyg38          0          5 OPT_ESTIMATE (@"SEL$683B0107" JOIN ("I"@"SEL$2" "P
                                     "@"SEL$2") ROWS=70.000000 )

1dch7djzpyg38          0          6 OPT_ESTIMATE (@"SEL$2" TABLE "P"@"SEL$2" ROWS=101.
                                     000000 )

1dch7djzpyg38          0          7 OPT_ESTIMATE (@"SEL$1" TABLE "C"@"SEL$1" ROWS=118.
                                     000000 )
```

| Name | Starts | E-Rows | A-Rows |
|---|---|---|---|
| | 1 | | 733 |
| | 1 | 733 | 733 |
| CUSTOMERS | 1 | 118 | 118 |
| | 1 | 936 | 936 |
| ORDERS | 1 | 840 | 840 |
| | 1 | 936 | 936 |
| | 1 | 70 | 70 |
| VW_NSO_1 | 1 | 70 | 70 |
| | 1 | 70 | 70 |
| PRODUCT | 1 | 101 | 101 |
| INVENTORIES | 1 | 1080 | 1069 |
| PRODUCT | 1 | 288 | 288 |
| ORDER_ITEMS | 1 | 2526 | 2526 |

```
SQL> @shared_proc
Enter value for 1: 1dch7djzpyg38

SQL_TEXT                       = select /*+ GATHER_PLAN_STATISTICS */
                                 c.cust_first_name, c.cust_last_name,
                                 ...cut for brevity...
SQL_ID                         = 1dch7djzpyg38
ADDRESS                        = 0000000071B12608
CHILD_ADDRESS                  = 0000000071FB1A48
CHILD_NUMBER                   = 0
USE_FEEDBACK_STATS             = Y
REASON                         = ...<reason>Auto Reoptimization Mismatch(1)
-----------------------------------------------------
SQL_TEXT                       = select /*+ GATHER_PLAN_STATISTICS */
                                 c.cust_first_name, c.cust_last_name
                                 ...cut for brevity...
SQL_ID                         = 1dch7djzpyg38
ADDRESS                        = 0000000071B12608
CHILD_ADDRESS                  = 000000007365FDC0
CHILD_NUMBER                   = 1
REASON                         =
-----------------------------------------------------
```

# Performance Feedback

solarwinds

- Automatically improves the degree of parallelism
  - Init.ora parameter,  PARALLEL_DEGREE_POLICY = ’ADAPTIVE’
- On 1st execution,  the optimizer decides
  - Whether to execute the statement in parallel
  - The degree of parallelism based on estimates
- After 1st execution, optimizer compares
  - Estimates with actual performance statistics
    - e.g. CPU Time
    - i.e. PARALLEL_MIN_TIME_THRESHOLD
  - If significantly different, the statement
    - Cursor is marked for reparsing
    - New execution statistics are stored as feedback
- Following executions use the performance feedback to determine DOP
  - If PARALLEL_DEGREE_POLICY not set, statistics feedback may change DOP

# Performance Feedback – Strange Behavior

- OPTIMIZER_ADAPTIVE_STATISTICS
  - Set to FALSE by default
  - Can set to TRUE to enable adaptive parallel distribution
    - Dynamic sampling for parallel query
    - Statistics feedback
    - Performance feedback
- PARALLEL_DEGREE_POLICY
  - Set to MANUAL by default
  - Must be ADAPTIVE
    - To enable Performance feedback
- Check hidden parameter _OPTIMIZER_PERFORMANCE_FEEDBACK
  - Must be set to ALL
  - http://www.oaktable.net/content/activating-and-deactivating-performance-feedback

```
optimizer_adaptive_statistics parallel_degree_policy _optimizer_performance_feedback
----------------------------- ---------------------- ------------------------------
FALSE                         ADAPTIVE               ALL
TRUE                          ADAPTIVE               ALL

parallel_degree_policy optimizer_adaptive_statistics _optimizer_performance_feedback
---------------------- ----------------------------- ------------------------------
ADAPTIVE               TRUE                          ALL
```

# Performance Feedback

```
select  /*+ GATHER_PLAN_STATISTICS PARALLEL(10) */
O_ID,O_C_ID,OL_AMOUNT,OL_QUANTITY from orders o, order_line ol where
ol_w_id = 2 and ol_d_id = 1 and ol_o_id = o_id and ol_w_id = o_w_id and
ol_d_id = o_d_id

Plan hash value: 2932861528
-------------------------------------------------------------------------------
|   Id  | Operation                     | Name       | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |            |      1 |        |      3 |
|     1 |  PX COORDINATOR               |            |      1 |        |      3 |
|     2 |   PX SEND QC (RANDOM)         | :TQ10002   |      0 |      3 |      0 |
|   *  3 |    HASH JOIN BUFFERED         |            |      0 |      3 |      0 |
|     4 |     JOIN FILTER CREATE        | :BF0000    |      0 |      3 |      0 |
|-    5 |      NESTED LOOPS             |            |      0 |      3 |      0 |
|     6 |       BUFFER SORT             |            |      0 |        |      0 |
|     7 |        PX RECEIVE             |            |      0 |        |      0 |
|     8 |         PX SEND HASH          | :TQ10000   |      0 |        |      0 |
|-    9 |          STATISTICS COLLECTOR |            |      1 |        |    159K|
|   * 10 |           INDEX RANGE SCAN    | ORDERS_I2  |      1 |    130K|    159K|
|- * 11 |       INDEX RANGE SCAN        | IORDL      |      0 |   4313K|      0 |
|    12 |     PX RECEIVE                |            |      0 |   4313K|      0 |
|    13 |      PX SEND HASH             | :TQ10001   |      0 |   4313K|      0 |
|    14 |       JOIN FILTER USE         | :BF0000    |      0 |   4313K|      0 |
|    15 |        PX BLOCK ITERATOR      |            |      0 |   4313K|      0 |
|   * 16 |         INDEX FAST FULL SCAN  | IORDL      |      0 |   4313K|      0 |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   3 - access("OL_O_ID"="O_ID" AND "OL_W_ID"="O_W_ID" AND "OL_D_ID"="O_D_ID")
  10 - access("O_W_ID"=2 AND "O_D_ID"=1)
  11 - access("OL_W_ID"=2 AND "OL_D_ID"=1 AND "OL_O_ID"="O_ID")
  16 - access(:Z>=:Z AND :Z<=:Z)
       filter(("OL_D_ID"=1 AND "OL_W_ID"=2))

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - Degree of Parallelism is 10 because of hint
   - statistics feedback used for this statement
   - performance feedback used for this statement
   - this is an adaptive plan (rows marked '-' are inactive)
```

```
select tq_id , cast(server_type as varchar2(10)) as server_type
      , instance , cast(process as varchar2(8)) as process, num_rows
      , round(ratio_to_report(num_rows)
            over (partition by dfo_number, tq_id, server_type) * 100) as "%"
      , cast(rpad('#', round(num_rows * 10 / nullif(max(num_rows)
           over (partition by dfo_number, tq_id, server_type), 0)), '#') as varchar2(10)) as
graph
      , round(bytes / 1024 / 1024) as mb
      , round(bytes / nullif(num_rows, 0)) as "bytes/row"
from  v$pq_tqstat
order by dfo_number , tq_id , server_type desc , instance , process;
```

| TQ_ID | SERVER_TYP | INSTANCE | PROCESS | NUM_ROWS | % | GRAPH | MB | bytes/row |
|---|---|---|---|---|---|---|---|---|
| 0 | Producer | 1 | P00A | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00B | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00C | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00D | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00E | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00F | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00G | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00H | 0 | 0 | | 0 | |
| 0 | Producer | 1 | P00I | 3 | 100 | ########## | 0 | 96 |
| 0 | Producer | 1 | P00J | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P000 | 1 | 33 | ########## | 0 | 256 |
| 0 | Consumer | 1 | P001 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P002 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P003 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P004 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P005 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P006 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P007 | 0 | 0 | | 0 | |
| 0 | Consumer | 1 | P008 | 1 | 33 | ########## | 0 | 256 |
| 0 | Consumer | 1 | P009 | 1 | 33 | ########## | 0 | 256 |
| 1 | Producer | 1 | P000 | 1 | 33 | ########## | 0 | 45 |
| 1 | Producer | 1 | P001 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P002 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P003 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P004 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P005 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P006 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P007 | 0 | 0 | | 0 | |
| 1 | Producer | 1 | P008 | 1 | 33 | ########## | 0 | 45 |
| 1 | Producer | 1 | P009 | 1 | 33 | ########## | 0 | 45 |
| 1 | Consumer | 1 | QC | 3 | 100 | ########## | 0 | 101 |

# SQL Plan Directives

- Are additional instructions for missing column group statistics or histograms
- The optimizer performs dynamic sampling on directive
  - Until statistics are gathered for the column group or extension
- Not tied to a specific sql statement – defined on a query expression
  - Can be used by similar queries (e.g.  city, state, zip in where clause)
- Are created in shared_pool & periodically written to the SYSAUX tablespace
  - DBA_SQL_PLAN_DIRECTIVES
  - DBA_SQL_PLAN_DIR_OBJECTS
  - DBMS_SPD package

```
SELECT TO_CHAR(d.directive_id) dir_id,
o.owner, o.object_name, o.subobject_name col_name,
o.object_type, d.type,d.state,d.reason
FROM dba_sql_plan_directives d, dba_sql_plan_dir_objects o
WHERE d.directive_id = o.directive_id
AND o.owner IN ('SOE') ORDER BY 1,2,3,4,5;
```

| DIR_ID | OWNER | OBJECT_NAM | COL_NAME | OBJECT_TYP | TYPE | STATE | REASON |
|---|---|---|---|---|---|---|---|
| 10725219980802248772 | SOE | ORDERS | | TABLE | DYNAMIC_SAMPLING | USABLE | JOIN CARDINALITY MISESTIMATE |
| 10725219980802248772 | SOE | ORDER_LINE | | TABLE | DYNAMIC_SAMPLING | USABLE | JOIN CARDINALITY MISESTIMATE |
| 11893362961552330 26 | SOE | ORDER_LINE | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |
| 12025048572496508512 | SOE | ORDERS | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |
| 14194012961398823517 | SOE | ORDERS | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |
| 57189726988536087 06 | SOE | ORDERS | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |
| 57189726988536087 06 | SOE | ORDER_LINE | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |
| 92975179242363766 0 | SOE | ORDER_LINE | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |
| 97034378650779572 5 | SOE | ORDERS | | TABLE | DYNAMIC_SAMPLING_RESULT | USABLE | VERIFY CARDINALITY ESTIMATE |

# SQL Plan Directives

- Must enable adaptive statistics
- New directive type in 12.2 - DYNAMIC_SAMPLING_RESULTS
  - Results are stored for future use
- Need to turn on feature to  automatically create extended statistics
  - exec DBMS_STATS.SET_PARAM ('AUTO_STAT_EXTENSIONS','ON');

```
       DIRECTIVE_ID TYPE                    STATE      REASON                                NOTES
------------------- ----------------------- ---------- ------------------------------------- ----------------------------------------
10125300680606620270 DYNAMIC_SAMPLING        USABLE     SINGLE TABLE CARDINALITY MISESTIMATE <spd_note>
                                                          <internal_state>MISSING_STATS</interna
                                                        l_state>
                                                          <redundant>NO</redundant>
                                                          <spd_text>{E(TEST.T_ONTIME)[ORIGIN_AIR
                                                        PORT_ID, ORIGIN_CITY_MARKET_ID]}</spd_te
                                                        xt>
                                                        </spd_note>

 4047921719485763684 DYNAMIC_SAMPLING_RESULT USABLE     VERIFY CARDINALITY ESTIMATE          <spd_note>
                                                          <internal_state>NEW</internal_state>
                                                          <redundant>NO</redundant>
                                                          <spd_text>{(TEST.T_ONTIME, num_rows=67
                                                        75800) - (SQL_ID:cur722y6p9fgs, Stopped,
                                                         T.CARD=31[-2 -1])}</spd_text>
                                                        </spd_note>

13326698477835124075 DYNAMIC_SAMPLING_RESULT USABLE     VERIFY CARDINALITY ESTIMATE          <spd_note>
                                                          <internal_state>NEW</internal_state>
                                                          <redundant>NO</redundant>
                                                          <spd_text>{(TEST.T_ONTIME, num_rows=67
                                                        75800) - (SQL_ID:5t18ywgrh6hdc, T.CARD=5
                                                        279[-2 -2])}</spd_text>
                                                        </spd_note>
```

```
NOTES
----------------------------------------
<spd_note>
  <internal_state>HAS_STATS</internal_st
ate>
  <redundant>NO</redundant>
  <spd_text>{E(TEST.T_ONTIME)[ORIGIN_AIR
PORT_ID, ORIGIN_CITY_MARKET_ID]}</spd_te
xt>
</spd_note>
```

# 12c Statistic Gathering

solarwinds

- Automatically gathered
  - Index, insert append, CTAS

```
SQL> CREATE TABLE ord (id NUMBER, cust_name VARCHAR2(30), total NUMBER);

Table created.

SQL> INSERT INTO ord
  2  SELECT ROWNUM, 'Customer_First_Middle_Last_Nam', MOD(ROWNUM,10)
  3  FROM dual connect by level <=1000000;

1000000 rows created.

SQL> SELECT table_name, num_rows, blocks, avg_space, avg_row_len, last_analyzed
  2  FROM dba_tables WHERE table_name='ORD';

TABLE_NAME             NUM_ROWS     BLOCKS  AVG_SPACE AVG_ROW_LEN LAST_ANALYZED
-------------------- ---------- ---------- ---------- ----------- --------------
ORD

1 row selected.

SQL> CREATE INDEX ord_total_idx ON Ord(total);

Index created.

SQL> SELECT index_name, num_rows, blevel, leaf_blocks, distinct_keys
  2  FROM dba_indexes WHERE index_name = 'ORD_TOTAL_IDX';

INDEX_NAME             NUM_ROWS     BLEVEL LEAF_BLOCKS DISTINCT_KEYS
-------------------- ---------- ---------- ----------- -------------
ORD_TOTAL_IDX           1000000          2        1939            10

1 row selected.
```

```
SQL> CREATE TABLE ord2 (id NUMBER, cust_name VARCHAR2(30), total NUMBER);

Table created.

SQL> CREATE INDEX ord2_total_idx ON ord2(total);

Index created.

SQL> INSERT /*+ APPEND */ INTO ord2 SELECT ROWNUM, 'Customer_First_Middle_Last_Nam',
  2  MOD(ROWNUM,10) FROM dual CONNECT BY LEVEL <=1000000;

1000000 rows created.

SQL> SELECT table_name, num_rows, blocks, avg_space, avg_row_len, last_analyzed
  2  FROM dba_tables WHERE table_name='ORD2';

TABLE_NAME             NUM_ROWS     BLOCKS  AVG_SPACE AVG_ROW_LEN LAST_ANALYZED
-------------------- ---------- ---------- ---------- ----------- -------------------
ORD2                    1000000       6203          0          39 01/19/2018 13:58:18

1 row selected.

SQL> SELECT column_name, num_distinct, density, histogram, notes
  2  FROM dba_tab_col_statistics WHERE table_name='ORD2';

COLUMN_NAME          NUM_DISTINCT    DENSITY HISTOGRAM       NOTES
-------------------- ------------ ---------- --------------- ------------------------------
TOTAL                          10         .1 NONE            STATS_ON_LOAD
CUST_NAME                       1          1 NONE            STATS_ON_LOAD
ID                        1000000   .000001 NONE            STATS_ON_LOAD

3 rows selected.

SQL> SELECT index_name, num_rows, blevel, leaf_blocks, distinct_keys
  2  FROM dba_indexes WHERE index_name = 'ORD2_TOTAL_IDX';

INDEX_NAME             NUM_ROWS     BLEVEL LEAF_BLOCKS DISTINCT_KEYS
-------------------- ---------- ---------- ----------- -------------
ORD2_TOTAL_IDX                0          0           0             0

1 row selected.

SQL> EXEC DBMS_STATS.GATHER_TABLE_STATS -
>    (ownname =>'soe',tabname=>'ORD2',options=>'GATHER AUTO');

SQL> SELECT index_name, num_rows, blevel, leaf_blocks, distinct_keys
  2  FROM dba_indexes WHERE index_name = 'ORD2_TOTAL_IDX';

INDEX_NAME             NUM_ROWS     BLEVEL LEAF_BLOCKS DISTINCT_KEYS
-------------------- ---------- ---------- ----------- -------------
ORD2_TOTAL_IDX          1000000          2        1739            10
```

# Understand Statistics Gathering

- GATHER_*_STATS procedures have many parameters
  - Consider taking the default values
  - exec dbms_stats.gather_schema_stats('SOE');
- New 12.2 optimizer statistics advisor
  - Based on 23 predefined rules
    - V$STATS_ADVISOR_RULES
    - Run using DBMS_STATS
      - View tasks in DBA_ADVISOR_TASKS
  - Makes recommendations on collecting stats
  - Can generate scripts for statistics gathering
    - Uses statistic gathering best practices
  - Advisor scripts on next slides

DBMS_STATS package
- Rewritten in 11g
  - A Faster & better AUTO_SAMPLE_SIZE
  - 100% in less time & more accurate than 10% estimate
- Avoid using ESTIMATE_PERCENT

GET_PREFS function

```
select
dbms_stats.get_prefs('PREFERENCE_OVERRIDES_PARAMETER')
from dual;
```

| | |
|---|---|
| AUTO_STAT_EXTENSIONS | ON |
| AUTOSTATS_TARGET | AUTO |
| CASCADE | DBMS_STATS.AUTO_CASCADE |
| CONCURRENT | OFF |
| DEGREE | NULL |
| ESTIMATE_PERCENT | DBMS_STATS.AUTO_SAMPLE_SIZE |
| GLOBAL_TEMP_TABLE_STATS | SESSION |
| GRANULARITY | AUTO |
| INCREMENTAL | FALSE |
| INCREMENTAL_LEVEL | PARTITION |
| INCREMENTAL_STALENESS | ALLOW_MIXED_FORMAT |
| METHOD_OPT | FOR ALL COLUMNS SIZE AUTO |
| NO_INVALIDATE | DBMS_STATS.AUTO_INVALIDATE |
| OPTIONS | GATHER |
| PUBLISH | TRUE |
| STALE_PERCENT | 10 |
| TABLE_CACHED_BLOCKS | 1 |

# Optimizer Statistics Advisor

solarwinds

## 1. Create task

```
EXEC DBMS_STATS.DROP_ADVISOR_TASK('STAT_ADVICE');

DECLARE
  task_name VARCHAR2(100);
  results VARCHAR2(32767);
BEGIN
  task_name := 'STAT_ADVICE';
  results := DBMS_STATS.CREATE_ADVISOR_TASK(task_name);
END;
/

select task_name, advisor_name, created, status from
dba_advisor_tasks where advisor_name = 'Statistics Advisor';
```

## 2. Define filters>

```
  filter1 CLOB;  -- disable advisor on all objects
  filter2 CLOB; -- enable advice on SOE.ORDER_LINE
  filter3 CLOB; --  disable rule AvoidDropRecreate
  filter4 CLOB; -- enable rule UseGatherSchemaStats
BEGIN
filter1 := DBMS_STATS.CONFIGURE_ADVISOR_OBJ_FILTER(
  task_name => 'STAT_ADVICE',
  stats_adv_opr_type => 'EXECUTE',
  rule_name => NULL,
  ownname => NULL,
  tabname => NULL,
  action => 'DISABLE' );

filter2 := DBMS_STATS.CONFIGURE_ADVISOR_OBJ_FILTER(
  task_name => 'STAT_ADVICE',
  stats_adv_opr_type => 'EXECUTE',
  rule_name => NULL,
  ownname => 'SOE',
  tabname => 'ORDER_LINE',
  action => 'ENABLE' );

filter3 := DBMS_STATS.CONFIGURE_ADVISOR_RULE_FILTER(
  task_name => 'STAT_ADVICE',
  stats_adv_opr_type => 'EXECUTE',
  rule_name => 'AvoidDropRecreate',
  action => 'DISABLE' );

filter4 := DBMS_STATS.CONFIGURE_ADVISOR_RULE_FILTER(
  task_name => 'STAT_ADVICE',
  stats_adv_opr_type => 'EXECUTE',
  rule_name => 'UseGatherSchemaStats',
  action => 'ENABLE' );
END;
/
```

## 3. Execute task

```
DECLARE
  task_name VARCHAR2(100);
  results VARCHAR2(32767);
BEGIN
  task_name := 'STAT_ADVICE';
  results := DBMS_STATS.EXECUTE_ADVISOR_TASK(task_name);
END;
/
```

# Optimizer Statistics Advisor – Cont.

## 4. Report task

```
set pagesize 1000
set linesize 132
set long 1000000
select dbms_stats.report_advisor_task('STAT_ADVICE',null,'text','all','all') as report  from dual;
```

## 5. Generate script

```
VAR script CLOB
DECLARE
  task_name VARCHAR2(100);
BEGIN
  task_name := 'STAT_ADVICE';
  :script := DBMS_STATS.SCRIPT_ADVISOR_TASK(task_name);
END;
/
```

## 6. Display script>

```
set linesize 132
set long 100000
set pagesize 0
set longchunksize 100000
set serveroutput on

DECLARE
  v_len NUMBER(10);
  v_offset NUMBER(10) :=1;
  v_amount NUMBER(10) :=10000;
BEGIN
  v_len := DBMS_LOB.getlength(:script);
  WHILE (v_offset < v_len)
  LOOP

DBMS_OUTPUT.PUT_LINE(DBMS_LOB.SUBST
R(:script,v_amount,v_offset));
    v_offset := v_offset + v_amount;
  END LOOP;
END;
/
```

# Optimizer Statistics Advisor Report

```
REPORT
--------------------------------------------------------------------------------
GENERAL INFORMATION
--------------------------------------------------------------------------------

 Task Name        : STAT_ADVICE
 Execution Name   : EXEC_611
 Created: 02-05-18 10:41:33
 Last Modified    : 02-05-18 10:51:58
--------------------------------------------------------------------------------
SUMMARY
--------------------------------------------------------------------------------
 For execution EXEC_611 of task STAT_ADVICE, the Statistics Advisor has 2
 finding(s). The findings are related to the following rules:
 AVOIDSETPROCEDURES, USEDEFAULTPARAMS. Please refer to the finding section for
 detailed information.
--------------------------------------------------------------------------------
FINDINGS
--------------------------------------------------------------------------------
 Rule Name:AvoidSetProcedures
 Rule Description:  Avoid Set Statistics Procedures
 Finding:  There are 11 SET_[COLUMN|INDEX|TABLE|SYSTEM]_STATS procedures being
  used for statistics gathering.

 Operation:
 set_table_stats(tabname=>'WAREHOUSE', numrows=>2, numblks=>5, avgrlen=>88, flags=>6);
 set_table_stats(tabname=>'STOCK', numrows=>200000, numblks=>9077, avgrlen=>306, flags=>6);
 set_table_stats(tabname=>'SQLSAT_IND', numrows=>2473, numblks=>80, avgrlen=>107, flags=>6);
 set_table_stats(tabname=>'SQLSAT_CNT', numrows=>107, numblks=>5, avgrlen=>89, flags=>6);
 set_table_stats(tabname=>'ORDER_LINE', numrows=>61031984, numblks=>0, avgrlen=>63, flags=>6);
 set_table_stats(tabname=>'ORDERS', numrows=>6103866, numblks=>29477, avgrlen=>31, flags=>6);
 set_table_stats(tabname=>'NEW_ORDER', numrows=>181977, numblks=>0, avgrlen=>11,flags=>6);
 set_table_stats(tabname=>'ITEM', numrows=>100000, numblks=>1126, avgrlen=>72, flags=>6);

REPORT
--------------------------------------------------------------------------------
 set_table_stats(tabname=>'HISTORY', numrows=>5318656, numblks=>36617, avgrlen=>44, flags=>6);
 set_table_stats(tabname=>'DISTRICT', numrows=>20, numblks=>20, avgrlen=>90, fla  gs=>6);
 set_table_stats(tabname=>'CUSTOMER', numrows=>42000, numblks=>3394, avgrlen=>576, flags=>6);

 Recommendation:  Do not use SET_[COLUMN|INDEX|TABLE|SYSTEM]_STATS procedures.
                  Gather statistics instead of setting them.

 Rationale:  SET_[COLUMN|INDEX|TABLE|SYSTEM]_STATS will cause bad plans due to
             wrong or inconsistent statistics.

--------------------------------------------------
```

```
--------------------------------------------------
Rule Name:UseDefaultParams

Rule Description:  Use Default Parameters in Statistics Collection Procedures

Finding:  There are 33 statistics operation(s) using nondefault parameters.

Operation:

gather_schema_stats(ownname=>'soe', estimate_percent=>1, method_opt=>'FOR ALL
       COLUMNS SIZE 1', gather_temp=>FALSE, gather_fixed=>FALSE);
delete_schema_stats(ownname=>'soe', stattype=>'ALL');
gather_table_stats(ownname=>'soe', tabname=>'orders', estimate_percent=>1,
       method_opt=>'FOR ALL COLUMNS SIZE 1');
gather_table_stats(ownname=>'soe', tabname=>'order_line', estimate_percent=>1,
       method_opt=>'FOR ALL COLUMNS SIZE 1');
 ...

Recommendation:  Use default parameters for statistics operations.

Example:
-- Gathering statistics for 'SH' schema using all default parameter values:
BEGIN dbms_stats.gather_schema_stats('SH'); END;
-- Also the non default parameters can be overriden by setting
'PREFERENCE_OVERRIDES_PARAMETER' preference.

-- Overriding non default parameters and preferences for all tables in the
system and to use dbms_stats for gathering statistics:
begin dbms_stats.set_global_prefs('PREFERENCE_OVERRIDES_PARAMETER', 'TRUE');
end;

-- Overriding non default parameters and preferences for 'SH.SALES':
begin dbms_stats.set_table_prefs('SH','SALES',
'PREFERENCE_OVERRIDES_PARAMETER', 'TRUE'); end;

Rationale:  Using default parameter values for statistics gathering operations
   is more efficient.
```

# Optimizer Statistics Advisor Script

```
-- Script generated for the recommendations from execution EXEC_989
-- in the statistics advisor task STAT_ADVICE
-- Script version 12.2
-- No scripts will be provided for the rule USEAUTOJOB.
       Please check the report for more details.
-- No scripts will be provided for the rule COMPLETEAUTOJOB.
-- No scripts will be provided for the rule MAINTAINSTATSHISTORY.

...cut for brevity...

-- Scripts for rule USECONCURRENT
-- Rule Description: Use Concurrent preference for Statistics Collection
-- Scripts for rule USEDEFAULTPREFERENCE
-- Rule Description: Use Default Preference for Stats Collection
-- Scripts for rule USEDEFAULTOBJECTPREFERENCE
-- Rule Description: Use Default Object Preference for statistics collection
-- Setting object-level preferences to default values
-- setting CASCADE to default value for object level preference
-- setting ESTIMATE_PERCENT to default value for object level preference
-- setting METHOD_OPT to default value for object level preference
-- setting GRANULARITY to default value for object level preference
-- setting NO_INVALIDATE to default value for object level preference

-- Scripts for rule USEINCREMENTAL
-- Rule Description:
--    Statistics should be maintained incrementally when it is beneficial

begin dbms_stats.set_table_prefs('SH', 'COSTS', 'INCREMENTAL', 'TRUE'); end;
/
begin dbms_stats.set_table_prefs('SH','SALES', 'INCREMENTAL', 'TRUE'); end;
/
```

```
declare
  obj_filter_list dbms_stats.ObjectTab;
  obj_filter
dbms_stats.ObjectElem;
  obj_cnt         number := 0;
begin
  obj_filter_list(obj_cnt) := obj_filter;
  obj_filter.ownname := 'SH';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'PROMOTIONS';
  obj_filter_list.extend();
  obj_cnt := obj_cnt + 1;
  obj_filter.ownname := 'SOE';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'CUSTOMER';
  obj_filter_list.extend();
  obj_cnt := obj_cnt + 1;
  obj_filter_list(obj_cnt) := obj_filter;
  obj_filter.ownname := 'SOE';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'DISTRICT';
  obj_filter_list.extend();
  obj_filter_list(obj_cnt) := obj_filter;
  obj_filter.ownname := 'SOE';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'ITEM';
  obj_filter_list.extend();
  obj_cnt := obj_cnt + 1;
  obj_filter_list(obj_cnt) := obj_filter;
  dbms_stats.gather_database_stats(
    obj_filter_list=>obj_filter_list);
end;
/
```

```
declare
  obj_filter_list dbms_stats.ObjectTab;
  obj_filter      dbms_stats.ObjectElem;
  obj_cnt         number := 0;
begin
  obj_filter_list :=
dbms_stats.ObjectTab();
  obj_filter.ownname := 'SOE';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'ORDER_LINE';

obj_filter_list.extend();
  obj_cnt := obj_cnt + 1;
  obj_filter_list(obj_cnt) := obj_filter;
  obj_filter.ownname := 'SOE';

obj_filter_list(obj_cnt) := obj_filter;
  obj_filter.ownname  := 'SOE';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'STOCK';
  obj_filter_list.extend();
  obj_cnt := obj_cnt + 1;
  obj_filter_list(obj_cnt) := obj_filter;
  obj_filter.ownname := 'SOE';
  obj_filter.objtype := 'TABLE';
  obj_filter.objname := 'WAREHOUSE';
  obj_filter_list.extend();
  obj_cnt := obj_cnt + 1;

obj_filter_list(obj_cnt) := obj_filter;
  dbms_stats.gather_database_stats(
    obj_filter_list=>obj_filter_list);
end;
/
```

# SQL Plan Management (Baselines)

solarwinds

- 12.1 SPM/Baseline Changes
  - SPM evolve advisor is an Auto Task (SYS_AUTO_SPM_EVOLVE_TASK)
    - Runs nightly in maintenance window
    - Automatically runs the evolve process for non-accepted plans in SPM
    - DBA views results of nightly task using DBMS_SPM.REPORT_AUTO_EVOLVE_TASK
    - Can Manage via OEM or DBMS_AUTO_TASK_ADMIN
  - Still can manually evolve an unaccepted plan using OEM or DBMS_SPM
    - DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE has been deprecated but still there
- 12.2 New ability to limit which SQL statements are captured using filters

alter system set optimizer_capture_sql_plan_baselines=true;
exec dbms_spm.configure('AUTO_CAPTURE_PARSING_SCHEMA_NAME','SOE', true);

```
SQL> select PARAMETER_NAME,PARAMETER_VALUE from dba_sql_management_config;

PARAMETER_NAME                           PARAMETER_VALUE
---------------------------------------- ------------------------------------------------------------
SPACE_BUDGET_PERCENT                     10
PLAN_RETENTION_WEEKS                     53
AUTO_CAPTURE_PARSING_SCHEMA_NAME         parsing_schema IN (SOE) AND parsing_schema NOT IN (SYS)
AUTO_CAPTURE_MODULE
AUTO_CAPTURE_ACTION
AUTO_CAPTURE_SQL_TEXT
```

# SQL Plan Management

- 12.2 Can manually capture baselines from AWR
  - DBMS_SPM.LOAD_PLANS_FROM_AWR
    - Takes begin and end snapshots
- How SPM works with adaptive plans
  - If auto-capture is enabled
    - Only the final plan is captured in baseline
    - When unaccepted adaptive plans evolve, the optimizer considers all subplans
      - If 1.5x better than existing baseline, the plan is accepted
    - Accepted plans are never adaptive
- How SPM works with adaptive cursor sharing (ACS)
  - If auto-capture is enabled, only one plan will be accepted
    - Recommendation is to manually load and accept all possible plans
      - So ACS will work
      - Otherwise the cursor will not be marked bind sensitive
        - Because the baseline will prevent it

# Approximate Query Processing

solarwinds

- Used for approximate ‘count distinct’ values and adds percentile aggregation
- Allows for faster processing of large data sets
  - Not exact but usually within 95%+ range
- Three new parameters – alter system/session
  - approx_for_aggregation Default=FALSE
    - Can be overridden by the next 2 parameters
    - If true, sets approx_for_percentile=ALL
  - approx_for_count_distinct Default=FALSE
    - Overrides exact COUNT DISTINCT clause
  - approx_for_percentile Default=NONE
    - Overrides MEDIAN clause (PERCENTILE_CONT)
    - Values can be PERCENTILE_CONT, PERCENTILE_DISC, and ALL
- Can be used without any changes to existing code
  - Replaces exact functions with SQL functions that return approximate results

# New SQL functions in 12.2

solarwinds

- Approximate query functions
  - APPROX_COUNT_DISTINCT (Introduced in 12.1)
  - APPROX_COUNT_DISTINCT_DETAIL
  - APPROX_COUNT_DISTINCT_AGG
  - TO_APPROX_COUNT_DISTINCT
  - APPROX_MEDIAN
  - APPROX_PERCENTILE
  - APPROX_PERCENTILE_DETAIL
  - APPROX_PERCENTILE_AGG
  - TO_APPROX_PERCENTILE
- Also in 12.2, support for approximate query functions
  - For materialized views and subsequent query rewrites

# Approximate SQL Example

```
SQL> SELECT /*+ no_parallel */ COUNT(DISTINCT ol_o_id) FROM order_line;

COUNT(DISTINCTOL_O_ID)
----------------------
                776370

------------------------------------------------------------------------------------------
SQL_ID  98398namw9713, child number 0
-------------------------------------
SELECT /*+ no_parallel */ count(distinct ol_o_id) from order_line

Plan hash value: 3590512611
-------------------------------------------------------------------------------
| Id  | Operation               | Name     | Rows  | Bytes |TempSpc| Cost (
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT        |          |       |       |       |   522K
|   1 |  SORT AGGREGATE         |          |     1 |    13 |       |
|   2 |   VIEW                  | VW_DAG_0 |   779K|  9899K|       |   522K
|   3 |    HASH GROUP BY        |          |   779K|  3807K|  1126M|   522K
|   4 |     INDEX FAST FULL SCAN| IORDL    |    98M|   467M|       |   413K
-------------------------------------------------------------------------------
Note
-----
   - Degree of Parallelism is 1 because of hint
```

95% accurate

```
SQL> SELECT /*+ no_parallel */ APPROX_COUNT_DISTINCT(ol_o_id) FROM order_line;

APPROX_COUNT_DISTINCT(OL_O_ID)
------------------------------
                        750823

------------------------------------------------------------------------------------------
SQL_ID  d1ryacmumw5c1, child number 0
-------------------------------------
SELECT /*+ no_parallel */ approx_count_distinct(ol_o_id) from order_line

Plan hash value: 2038893387
-------------------------------------------------------------------------------------------
| Id  | Operation             | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |       |       |       |   413K(100)|          |
|   1 |  SORT AGGREGATE APPROX|       |     1 |     5 |            |          |
|   2 |   INDEX FAST FULL SCAN| IORDL |    98M|   467M|   413K  (1)| 00:00:17 |
-------------------------------------------------------------------------------------------
Note
-----
   - Degree of Parallelism is 1 because of hint
```

# Approximate SQL Example Without Changing Code

```
SQL>  ALTER SESSION SET approx_for_count_distinct = true;

Session altered.

SQL> show parameter approx

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
approx_for_aggregation               boolean     FALSE
approx_for_count_distinct            boolean     TRUE
approx_for_percentile                string      none

SQL> SELECT /*+ no_parallel */ COUNT(DISTINCT ol_o_id) from order_line
  2  ;

COUNT(DISTINCTOL_O_ID)
----------------------
                750823


-------------------------------------------------------------------------------
SQL_ID  fqbtb985nfccr, child number 0
-------------------------------------
SELECT /*+ no_parallel */ count(distinct ol_o_id) from order_line

Plan hash value: 2038893387
-------------------------------------------------------------------------------
| Id  | Operation             | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |       |       |       |   413K(100)|          |
|   1 |  SORT AGGREGATE APPROX|       |     1 |     5 |            |          |
|   2 |   INDEX FAST FULL SCAN| IORDL |    98M|   467M|   413K  (1)| 00:00:17 |
-------------------------------------------------------------------------------
```

```
SQL> show parameter approx

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
approx_for_aggregation               boolean     FALSE
approx_for_count_distinct            boolean     FALSE
approx_for_percentile                string      none

SQL> ALTER SESSION SET approx_for_aggregation = true;

Session altered.

SQL>  SELECT /*+ no_parallel */ COUNT(DISTINCT ol_o_id) FROM order_line;

COUNT(DISTINCTOL_O_ID)
----------------------
                776370

SQL> show parameter approx

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
approx_for_aggregation               boolean     TRUE
approx_for_count_distinct            boolean     FALSE
approx_for_percentile                string      ALL


SQL> ALTER SESSION SET approx_for_count_distinct = true;

SQL>  show parameter approx

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
approx_for_aggregation               boolean     TRUE
approx_for_count_distinct            boolean     TRUE
approx_for_percentile                string      ALL
```

Why is it exact?

Need to set both

# Approximate Percentile Example Without Changing Code

```
SQL> ALTER SESSION SET approx_for_aggregation = false;

Session altered.

SQL> show parameter approx

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
approx_for_aggregation               boolean     FALSE
approx_for_count_distinct            boolean     FALSE
approx_for_percentile                string      NONE

SQL> SELECT ol_d_id, MEDIAN(ol_amount) FROM order_line GROUP BY ol_d_id
  2* ORDER BY  MEDIAN(ol_amount);

   OL_D_ID MEDIAN(OL_AMOUNT)
---------- -----------------
         5            196.57
         9            197.36
        10            198.09
         7            204.33
         1            204.65
         8            206.86
         4             206.9
         2            207.53
         6            209.39
         3            211.51

10 rows selected.

Plan hash value: 1081856132

-------------------------------------------------------------------------------
| Id  | Operation             | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |       |       |       |   419K(100)|          |
|   1 |  SORT ORDER BY        |       |    10 |    80 |   419K  (2)| 00:00:17 |
|   2 |   SORT GROUP BY       |       |    10 |    80 |   419K  (2)| 00:00:17 |
|   3 |    INDEX FAST FULL SCAN| IORDL |    98M|   748M|   413K  (1)| 00:00:17 |
-------------------------------------------------------------------------------
```

```
SQL> alter session set approx_for_aggregation=TRUE;

Session altered.

SQL> show parameter approx

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
approx_for_aggregation               boolean     TRUE
approx_for_count_distinct            boolean     FALSE
approx_for_percentile                string      ALL

SQL> SELECT ol_d_id, MEDIAN(ol_amount) FROM order_line
  2  GROUP BY ol_d_id ORDER BY  MEDIAN(ol_amount);

   OL_D_ID MEDIAN(OL_AMOUNT)
---------- -----------------
         5            199.26
         9            197.98
        10            196.84
         7            204.56
         1            205.34
         8            208.59
         4            206.72
         2            208.85
         6            205.94
         3            212.27

10 rows selected.

Plan hash value: 1081856132

-------------------------------------------------------------------------------
| Id  | Operation             | Name  | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |       |       |       |   419K(100)|          |
|   1 |  SORT ORDER BY        |       |    10 |    80 |   419K  (2)| 00:00:17 |
|   2 |   SORT GROUP BY APPROX|       |    10 |    80 |   419K  (2)| 00:00:17 |
|   3 |    INDEX FAST FULL SCAN| IORDL |    98M|   748M|   413K  (1)| 00:00:17 |
-------------------------------------------------------------------------------
```

99.68% accurate

# Summary

solarwinds

- The 12.2 Optimizer is getting smarter and easier to control
  - Two new parameters
    - Optimizer_adaptive_plans (true)
      - New feature: bitmap index pruning
    - Optimizer_adaptive_statistics (false)
      - Controls SQL plan directives, statistics feedback for joins,
      - Performance feedback, dynamic sampling for parallel query
- Plans can and do change
  - V$SQL_SHARED_CURSOR can help find out the why
- New 12.2 optimizer statistics advisor can help fine-tune statistics gathering
- SQL plan management
  - Can filter which SQL gets captured for baselines
- Approximate query processing
  - Consider this for analytic queries

# Thank You!!!

# Resolve Performance Issues quickly—Free Trial

- Try ***Database Performance Analyzer*** FREE for 14 days
- Improve root cause of slow performance
  - Quickly identify root cause of issues that impact end-user response time
  - See historical trends over days, months, and years
  - Understand impact of VMware® performance
  - Agentless architecture with no dependence on Oracle Packs, installs in minutes

**www.solarwinds.com/dpa-download/**

# Appendix

- CBO trace or 10053 event

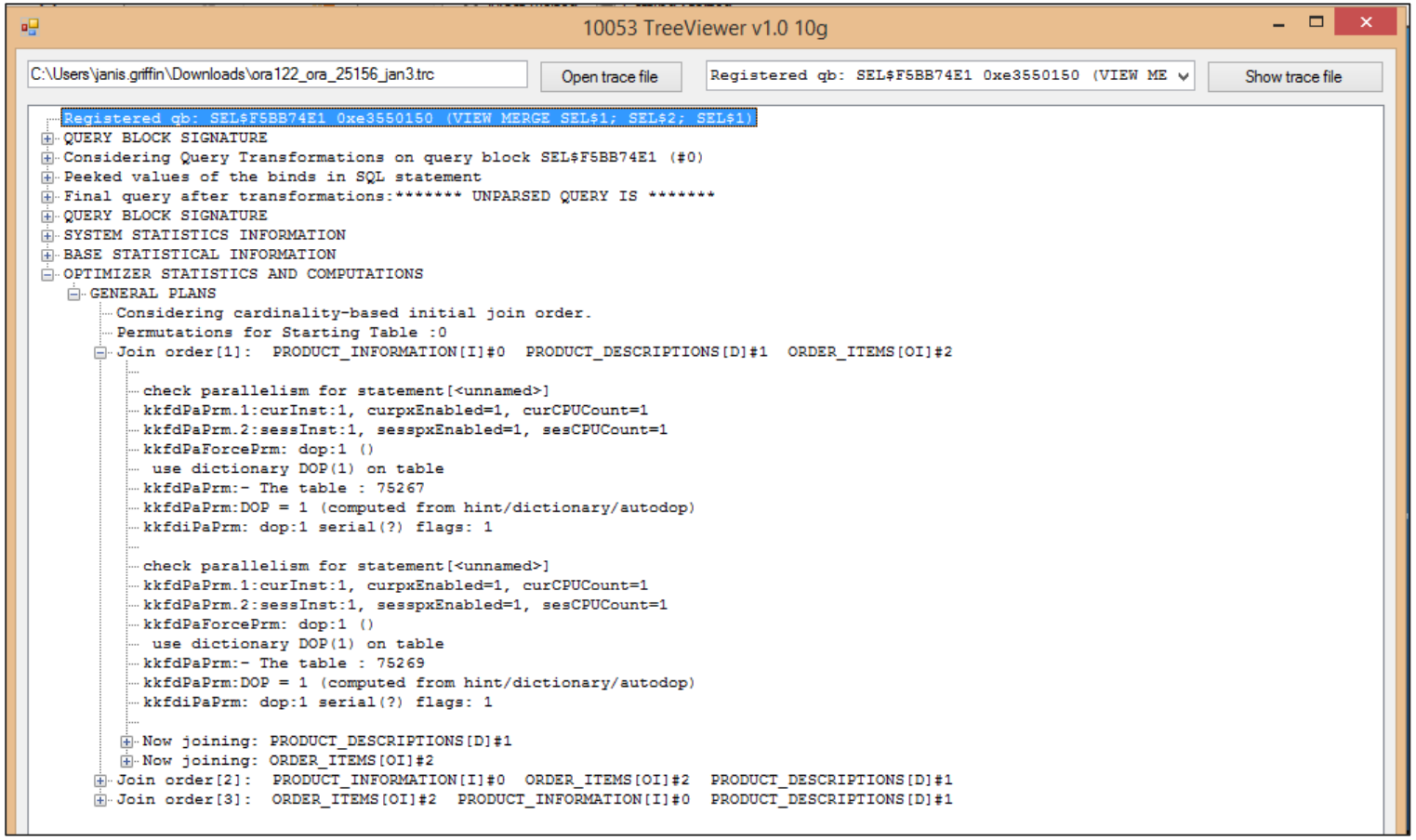
```
-- trc.sql
alter session set tracefile_identifier='&trc_name';
exec dbms_sqldiag.dump_trace(p_sql_id=>'&sql_id',p_child_number=>&child,p_component=>'Compiler',p_file_id=>'');
```

- Edit *&trc_name*.trc

```
----- Current SQL Statement for this session (sql_id=4rdftbf9srnmy) -----
/* SQL Analyze(77,0) */ select /*+ gather_plan_statistics  */ order_id, product_name
from products p, order_items oi
where p.product_id = oi.product_id
and unit_price < :b1
----- PL/SQL Stack -----
----- PL/SQL Call Stack -----
  object      line  object
  handle    number  name
0x735fd1d8       161  package body SYS.DBMS_SQLTUNE_INTERNAL.I_PROCESS_SQL_CALLOUT
0x735fd1d8     14190  package body SYS.DBMS_SQLTUNE_INTERNAL.I_PROCESS_SQL
0x737c8a60      1585  package body SYS.DBMS_SQLDIAG.DUMP_TRACE
0xbf72a730         1  anonymous block
*****************************************
Legend
The following abbreviations are used by optimizer trace.
CBQT - cost-based query transformation
JPPD - join predicate push-down
OJPPD - old-style (non-cost-based) JPPD
FPD - filter push-down
PM - predicate move-around
CVM - complex view merging
SPJ - select-project-join
SJC - set join conversion
SU - subquery unnesting
OBYE - order by elimination
OST - old style star transformation
ST - new (cbqt) star transformation
CNT - count(col) to count(*) transformation
JE - Join Elimination
JF - join factorization
CBY - connect by
SLP - select list pruning
DP - distinct placement
VT - vector transformation
AAT - Approximate Aggregate Transformation
...etc...
```

# Appendix

- 10053 Event TreeViewer UI (2011 but still works)
  - https://jonathanlewis.files.wordpress.com/2011/12/tvzip1.doc



# Appendix

solarwinds

```
--shared_proc.sql   --formats output from v$sql_shared_cursor

set serverout on size 1000000
Declare
 c  number;
 col_cnt   number;
 col_rec   dbms_sql.desc_tab;
 col_value varchar2(4000);
 ret_val   number;
Begin
 c := dbms_sql.open_cursor;
 dbms_sql.parse(c,'select q.sql_text, s.*  from v$sql_shared_cursor s, v$sql q where s.sql_id = q.sql_id
              and s.child_number = q.child_number and q.sql_id = ''&1''', dbms_sql.native);
 dbms_sql.describe_columns(c, col_cnt, col_rec);
 for idx in 1 .. Col_cnt loop
   dbms_sql.define_column(c, idx, col_value, 4000);
 end loop;
 ret_val := dbms_sql.execute(c);
 while(dbms_sql.fetch_rows(c) > 0) loop
   for idx in 1 .. Col_cnt loop
     dbms_sql.column_value(c, idx, col_value);
     if col_rec(idx).col_name in ('SQL_ID', 'ADDRESS', 'CHILD_ADDRESS','CHILD_NUMBER', 'SQL_TEXT', 'REASON') then
       dbms_output.put_line(rpad(col_rec(idx).col_name, 30) || ' = ' || col_value);
     elsif col_value = 'Y' then
       dbms_output.put_line(rpad(col_rec(idx).col_name, 30) || ' = ' || col_value);
     end if;
   end loop;
   dbms_output.put_line('-----------------------------------------------');
 end loop;
 dbms_sql.close_cursor(c);
End;
/
```

# Appendix

- 191 _optimizer_* hidden parameters
- OPTIMIZER_ADAPTIVE_PLANS control:
  - _OPTIMIZER_NLJ_HJ_ADAPTIVE_JOIN
  - _PX_ADAPTIVE_DIST_METHOD
  - _OPTIMIZER_STRANS_ADAPTIVE_PRUNING
- OPTIMIZER_ADAPTIVE_STATISTICS controls:
  - _OPTIMIZER_GATHER_FEEDBACK
  - _OPTIMIZER_USE_FEEDBACK
  - _OPTIMIZER_DSDIR_USAGE_CONTROL
  - _OPTIMIZER_USE_FEEDBACK_FOR_JOIN
  - _OPTIMIZER_ADS_FOR_PQ
- Many others not listed

```
-- hidden.sql
COLUMN ksppinm FORMAT A50
COLUMN ksppstvl FORMAT A50
SELECT
  ksppinm,
  ksppstvl
FROM
  x$ksppi a,
  x$ksppsv b
WHERE
  a.indx=b.indx
AND
  substr(ksppinm,1,1) = '_'
and ksppinm like '_opt%'
ORDER BY ksppinm;
```