solarwinds

DO YOU LACK STABILITY IN YOUR LIFE?

ESPECIALLY WHEN IT COMES TO ORACLE EXECUTION PLANS?

Janis Griffin
Senior DBA / Performance Evangelist

WHO AM I

Senior DBA / Performance Evangelist for SolarWinds

- **JanisGriffin@solarwindscom**
- **Twitter® - @DoBoutAnything**
- **Current – 25+ Years in Oracle®, SQL Server®, ASE, and now MySQL®**
- **DBA and Developer**

Specialize in Performance Tuning

Review Database Performance for Customers and Prospects

Common Question – How do I tune it?

AGENDA

History of Plan Stability

- **In the beginning**
- **Outlines, Profiles, Patches & Baselines**

How the optimizer works

Why & how do plans change

- **V$SQL_PLAN & V$SQL_SHARED_CURSOR**
  - Helps with finding out the ‘WHY’
  - XML Query of column ‘Other_XML’

Which features may impact the plan

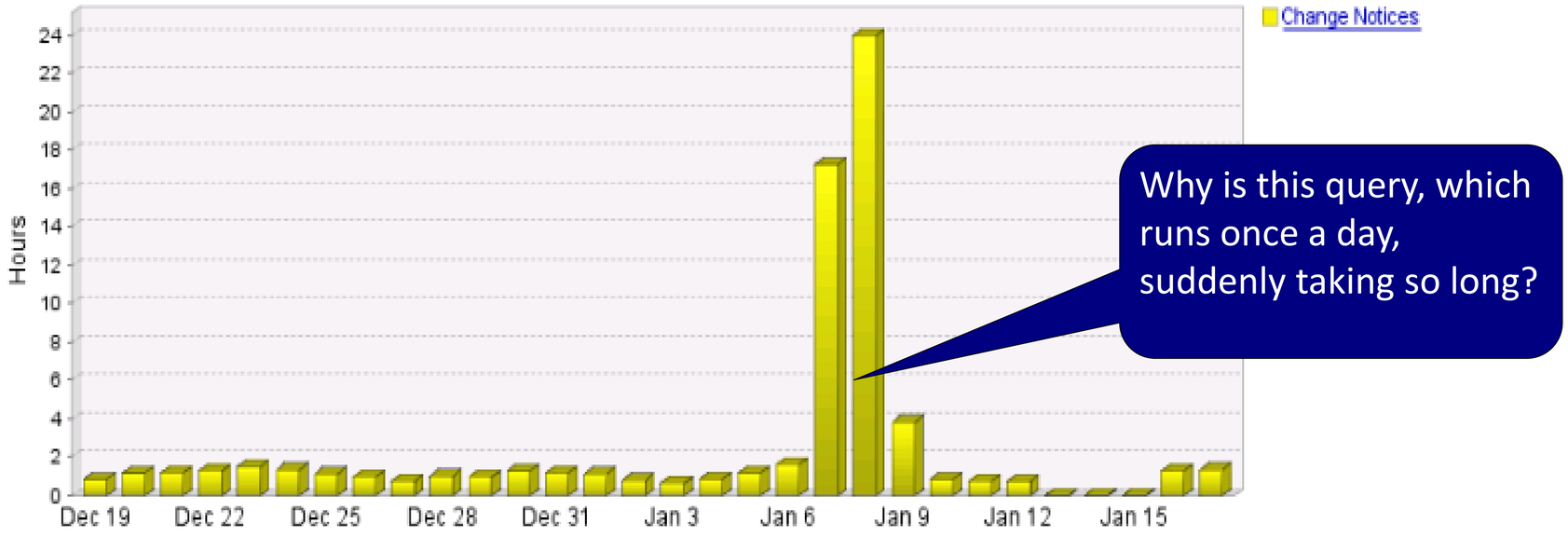
- **Adaptive Cursor Sharing**
- **SQL Directives & Statistics Feedback**

Using Baselines, Patches & Profiles to help stabilize plans

Q & A

QUERY SUDDENLY RUNS SLOWER

Specified SQL Statements | PROD_STA740
December 19, 2015 to January 17, 2016

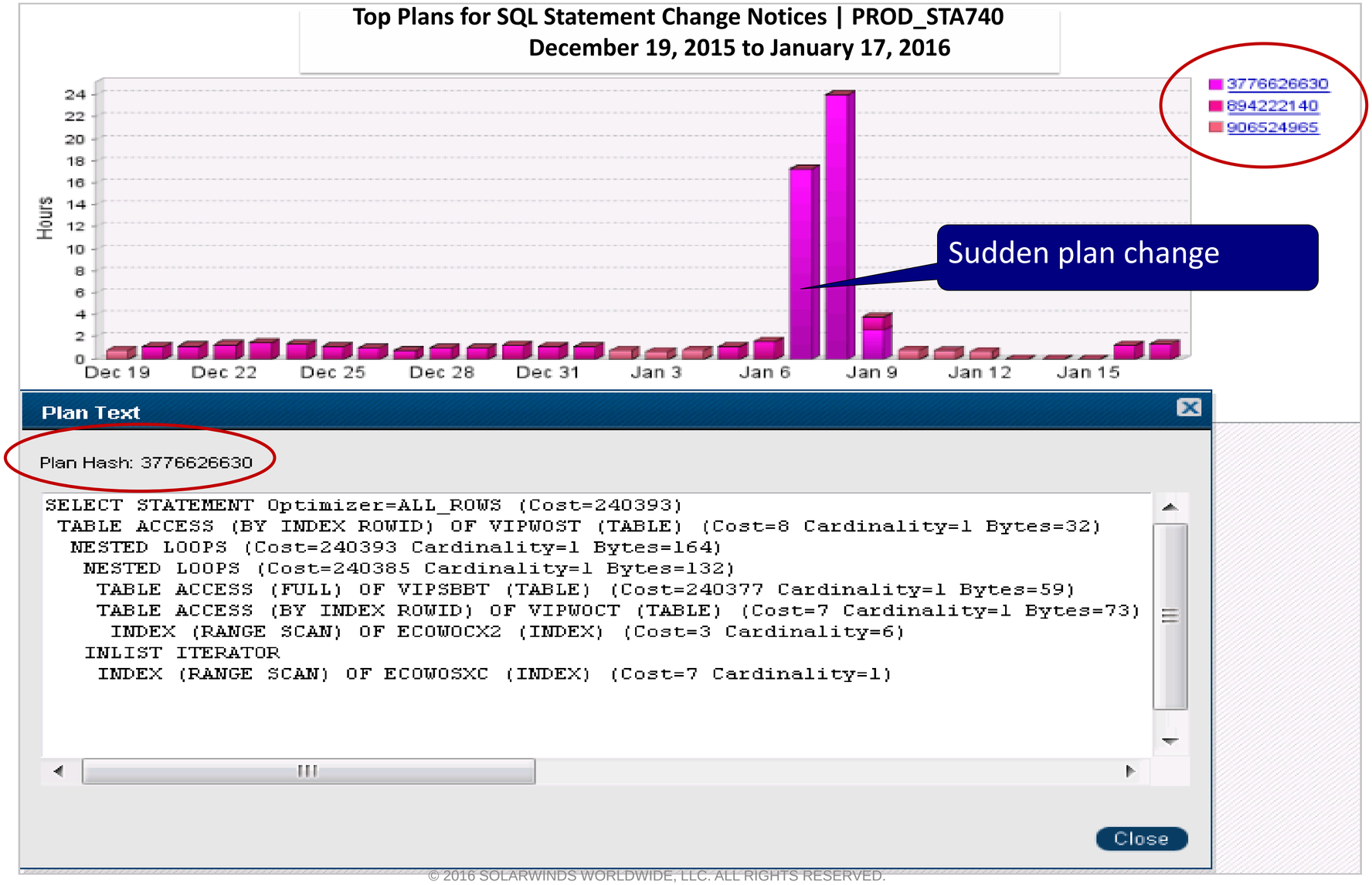


Hide Full SQL Text

Full SQL Text

SQL Hash Name	SQL Text
Change Notices	SELECT "A3"."ORIG_OPER_ID_WOC","A3"."DTE_ENT_WOC",TO_DATE("A3"."TME_ENT_WOC",'HH24:MI:SS'),"A3"."LS_CHG_DTE_WOC","A3"."SUB_ACCT_N FROM "WOC_BASE_COMP" "A3","WOS_SERV_CODE" "A2","SBB_BASE" "A1" WHERE "A3"."WO_KEY_WOC"="A2"."WO_KEY_WOS" AND "A3"."DTE_ENT_WOC"=TRUNC(SYSDATE@!-2) AND "A2"."SERV_NET_CHG_WOS"<>0 AND ("A3"."PRIN_WOC"=1000 OR "A3"."PRIN_WOC"=1100 OR "A3"."PRIN_WOC"=1200 OR "A3"."PRIN_WOC"=1300 OR "A3"."PRIN_WOC"=1400 OR "A3"."PRIN_WOC"=1500 OR "A3"."PRIN_WOC"=7100 OR "A3"."PRIN_WOC"=8000 OR "A3"."PRIN_WOC"=9000) AND ("A2"."SERV_CDE_WOS"='}J' OR "A2"."SERV_CDE_WOS"='}K' OR "A2"."SERV_CDE_WOS"='}L' OR "A2"."SERV_CDE_WOS"='}M') AND "A1"."SUB_ACCT_NO_SBB"="A3"."SUB_ACCT_NO_WOC"

PLAN REGRESSION



IN THE BEGINNING - HISTORY OF PLAN STABILITY

solarwinds

Oracle 8 – Introduced cost-based optimizer

- **Allowed for Hash joins, Histograms, Partitioning & Parallel queries**
- **Required statistics gathering**
  - Quickly found out that plans could change over time
- **8.1.7+ Stored Outlines to control plan changes**

Oracle 10g – SQL Profiles / Tuning Advisor

- **Sub-optimal execution plans still generated**
- **Performance Regression overtime - No Evolution**
- **DBMS_SQLTUNE – Costs $$$**

Oracle 11 – SQL Patches & SQL Plan Management (Baselines)

- **SQL Patches free both in Standard or Enterprise**
- **Baselines free with Enterprise**

Oracle 12C – Adaptive Optimizer

- **Allows for automatic plan evolution & SPM Evolve Advisor**

OUTLINES & PROFILES

solarwinds

Stored Outlines

- **Can ‘freeze’ a plan for a specific statement**
- **Used when sql changing between a couple of plans**
  - e.g. bind variable peeking
- **Implemented with hints**
  - So freeze is not absolutely guaranteed (e.g. hint uses index but index can be dropped)
  - DBMS_OUTLN / alter session set create_stored_outlines = true;

SQL Profiles

- **Created by SQL Tuning Advisor (dbms_sqltune - cost $$$)**
- **Similar to Outlines – implemented with hints**
- **Uses OPT_ESTIMATE hint – not always accurate**
  - Tries to improve cost estimates over time (factors 10x estimate – may be wrong)
- **Nightly look at SQLs to find better execution plan**

OUTLINES & PROFILES LIMITATIONS

Reactive versus Proactive

- **Performance issues have to occur before fix**

Depends on hints to limit optimizer choices

- **Not a guaranteed plan when changes happen**

Can grow stale over time

- **No evolution of plans as changes happen**

Outlines – Deprecated Announcement in 11

- **But still work in 12c**

Profiles/Tuning Advisor – Cost $$$

SQL PATCHES

SQL Patches is part of the SQL Repair Advisor (11g)

- **Used after a SQL statement fails with a critical error**
  - Example - index is dropped, the advisor recommends a patch using another index

DBA can manually add patches outside of SQL Repair Advisor

- **Used to insert hints into a query which can't be changed**
  - Useful for 3rd party applications
- **Uses the function: dbms_sqldiag_internal.i_create_patch**

Advantages over Profiles & Baselines

- **Advisor is FREE in both Standard & Enterprise Additions**
- **Works without tuning packs**

Limitations

- **If patch is used, it's often forgotten**
  - Need to periodically review performance (won't evolve)
- **Is more of a workaround than a solution**

SQL PLAN MANAGEMENT (BASELINES)

solarwinds

Preventative mechanism for Plan Stability

- **Optimizer records & evaluates execution plans over time (if run more than once)**
- **Baselines can evolve overtime for better performance**
- **Preserves performance regardless of changes**
- **DBA can verify that only comparable or better plans will be used**

Common Uses of Baselines

- **System & Data changes causing performance regressions**
- **Database Upgrades & New Application Installs**

In 12c

- **Manage SPM Evolve Advisor via OEM or DBMS_AUTO_TASK_ADMIN**
- **Still can evolve unaccepted plans manually using DBMS_SPM or OEM**

BASELINE LIMITATIONS

Baseline evolution is dependent on new plan performance

- **Must be >= 1.5x better**
- **Significant performance improvements could happen > 1x**
  - DBA must manually research and evolve these

Baselines are created using the signature of the SQL statement

- **Unique SQL identifier generated from the normalized SQL text**
  - Uncased & whitespaces removed
- **Could be wrong plan if same SQL statement is used in different schemas**
  - For example, missing index in one schema not the other
  - May result in many unaccepted baselines

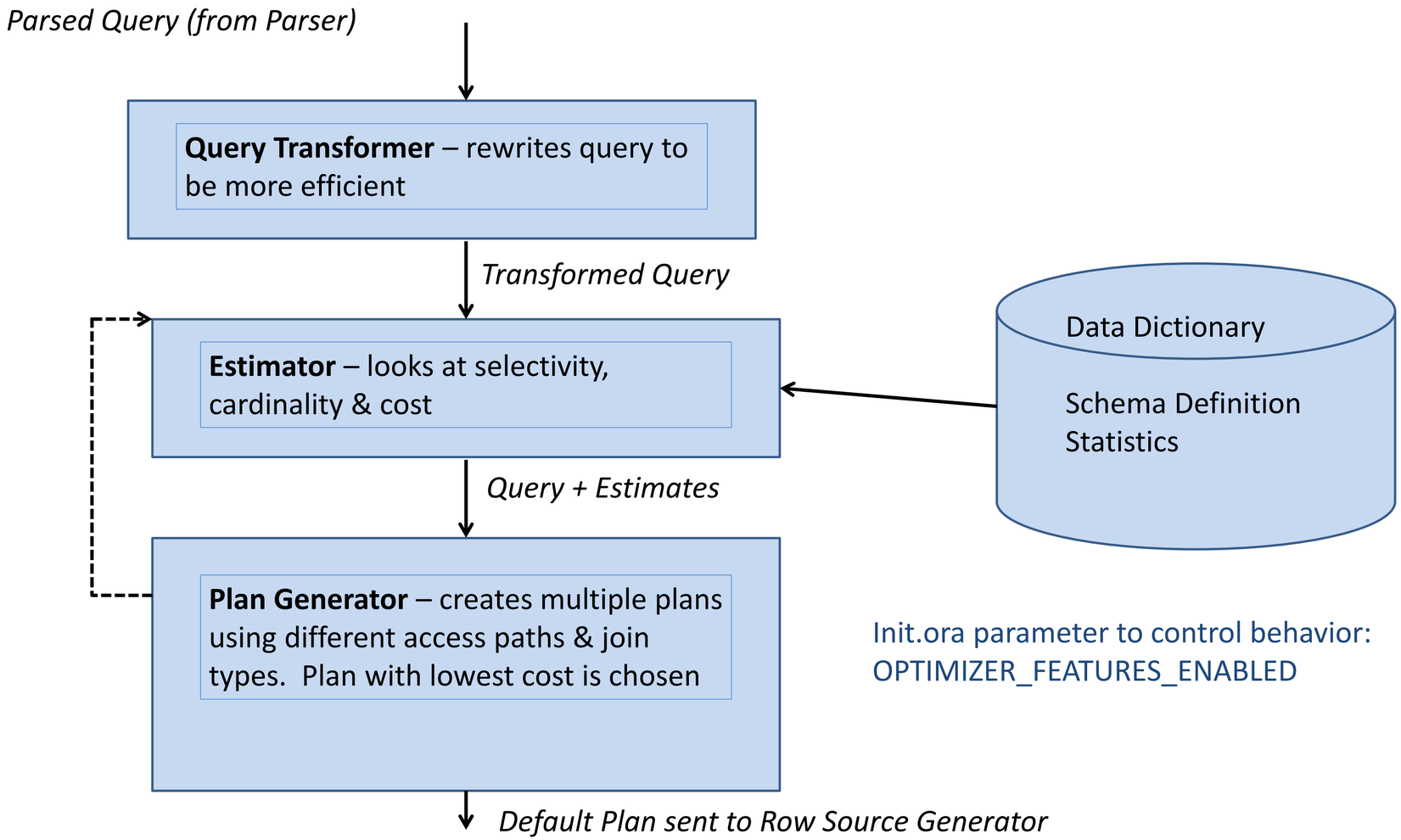
SQL statements using dblinks can’t be baselined

- **No remote access**

https://www.youtube.com/watch?v=g_I0DjpJ3a0

Hans Brinker Puts his Finger in the Dike

HOW THE OPTIMIZER WORKS



EXECUTION PLAN

Show the sequence of operations performed to run SQL Statement

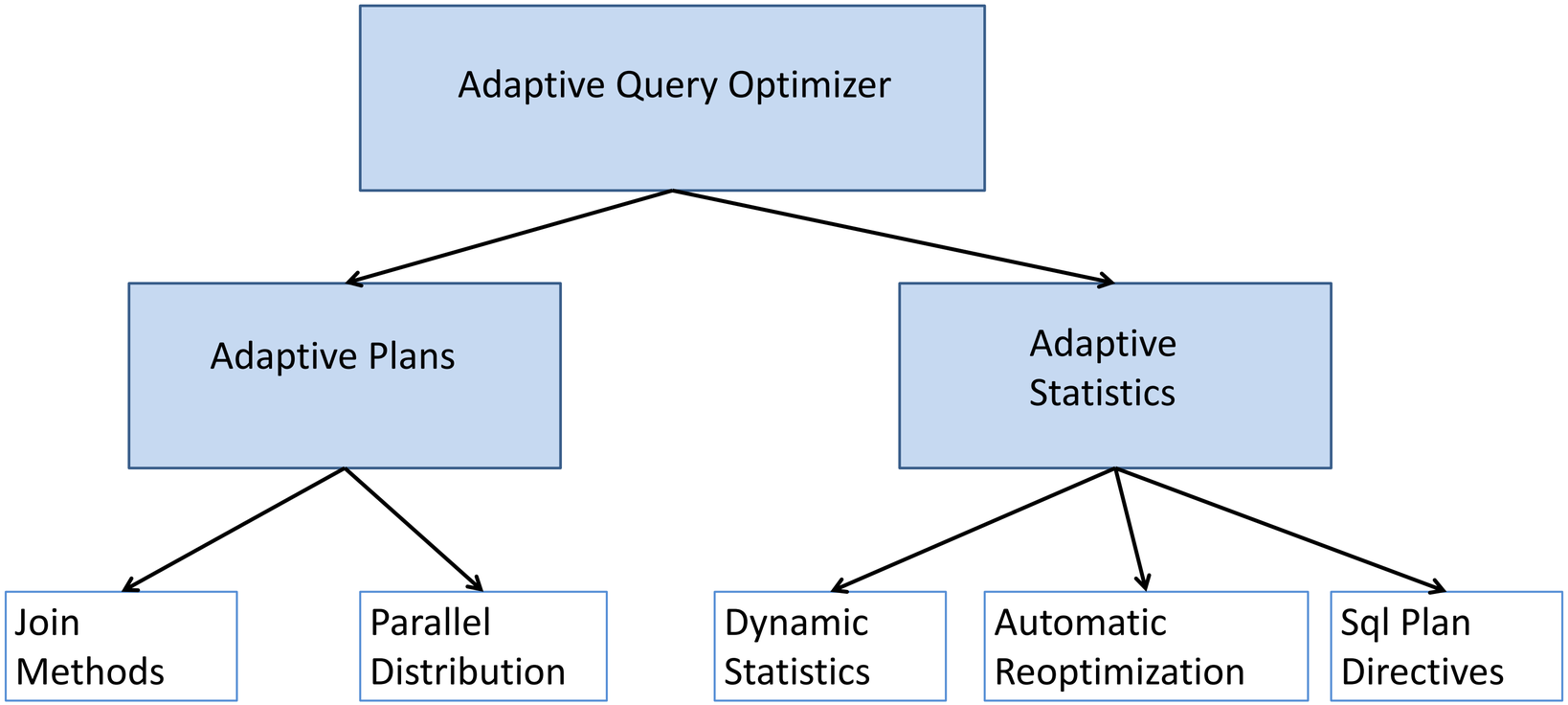
- **Order of the tables referenced in the statements**
- **Access method for each table in the statement**
  - INDEX
  - INLIST ITERATOR
  - TABLE ACCESS
  - VIEW
- **Join method in statement accessing multiple tables**
  - HASH JOIN
  - MERGE JOIN
  - NESTED LOOPS
- **Data manipulations**
  - CONCATENATION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

12C ADAPTIVE QUERY OPTIMIZER

Allows for run-time adjustments to execution plans

Can discover additional information

- **Which can lead to better statistics & optimal plans**



HOW TO CONTROL & VIEW ADAPTATIONS

solarwinds

Init.ora parameters that control Adaptive Plans

Name	Type	Value
optimizer_adaptive_features	boolean	TRUE
optimizer_adaptive_reporting_only	boolean	FALSE
optimizer_features_enable	string	12.1.0.1
optimizer_dynamic_sampling	integer	2

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

Use DBMS_XPLAN.DISPLAY_CURSOR to view Adaptations

- **Use format parameter '+report' for testing**
  - Works with 'reporting_only' mode

```
select * from table(dbms_xplan.display_cursor ('&sql_id',&child,format=>'+report'));
```

- **Use format parameter '+adaptive' to see all steps (active / inactive)**
  - Including instrumented statistics collectors

```
select * from table(dbms_xplan.display_cursor('&sql_id',&child,format=>'+adaptive'));
```

ADAPTIVE PLANS

```
SELECT sql_id, child_number,
            SUBSTR(sql_text, 1,30) sql_text,
             IS_RESOLVED_ADAPTIVE_PLAN,
             IS_REOPTIMIZABLE
  FROM v$sql
  WHERE sql_text like 'select /* jg */%'
  ORDER BY sql_id,child_number
```

```
select /* jg */ p.product_name
from order_items o, product p
where o.unit_price = :b1
and o.quantity > :b2
and o.product_id = p.product_id;
```

SQL_ID	CHILD_NUMBER	SQL_TEXT	IS_RESOLVED_ADAPTIVE	IS_REOPTIMIZABLE
8qpakg674n4mz	0	select /* jg */ p.product_name	Y	R
8qpakg674n4mz	1	select /* jg */ p.product_name	Y	Y
8qpakg674n4mz	2	select /* jg */ p.product_name	Y	N

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

OPTIMIZER_ADAPTIVE_REPORTING_ONLY

alter session set optimizer_adaptive_reporting_only=TRUE;

select * from table(dbms_xplan.display_cursor('8qpakg674n4mz',0,format=>'+report'));

```
SQL_ID  8qpakg674n4mz, child number 0
-------------------------------------
select /* jg */ p.product_name from order_items o, product p where
o.unit_price = :b1  and o.quantity > :b2  and o.product_id =
p.product_id

Plan hash value: 158447987
-------------------------------------------------------------------------------------
| Id  | Operation           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |              |       |       | 13184 (100)|          |
|*  1 |  HASH JOIN          |              |  1895 | 73905 | 13184   (3)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL | ORDER_ITEMS  |  1895 | 20845 | 11862   (3)| 00:00:01 |
|   3 |   TABLE ACCESS FULL | PRODUCT      |  1022K|    27M|  1314   (2)| 00:00:01 |
-------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("O"."PRODUCT_ID"="P"."PRODUCT_ID")
   2 - filter(("O"."UNIT_PRICE"=:B1 AND "O"."QUANTITY">:B2))
Note
-----
   - this is an adaptive plan

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  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |              |       |       | 13184 (100)|          |
|   1 |  NESTED LOOPS                 |              |       |       |            |          |
|   2 |   NESTED LOOPS                |              |  1895 | 73905 | 13184   (3)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL          | ORDER_ITEMS  |  1895 | 20845 | 11862   (3)| 00:00:01 |
|*  4 |    INDEX RANGE SCAN           | PRODUCT_IDX  |       |       |            |          |
|   5 |   TABLE ACCESS BY INDEX ROWID | PRODUCT      |     1 |    28 |  1314   (2)| 00:00:01 |
```

ACTUAL ADAPTIVE PLAN

Adapted on first execution

alter session set optimizer_adaptive_reporting_only=FALSE

```
SQL> select * from table(dbms_xplan.display_cursor('8qpakg674n4mz',1,format=>'+adaptive'));

SQL_ID  8qpakg674n4mz, child number 1
-------------------------------------
select /* jg */ p.product_name from order_items o, product p where
o.unit_price = :b1  and o.quantity > :b2  and o.product_id =
p.product_id

Plan hash value: 3627148456
-------------------------------------------------------------------------------------------------
|   Id  | Operation                     | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT              |              |       |       | 13184 (100)|          |
|- *  1 |  HASH JOIN                    |              |  1895 | 73905 | 13184   (3)| 00:00:01 |
|     2 |   NESTED LOOPS                |              |       |       |            |          |
|     3 |    NESTED LOOPS               |              |  1895 | 73905 | 13184   (3)| 00:00:01 |
|-    4 |     STATISTICS COLLECTOR      |              |       |       |            |          |
|   *  5 |      TABLE ACCESS FULL        | ORDER_ITEMS  |  1895 | 20845 | 11862   (3)| 00:00:01 |
|   *  6 |     INDEX RANGE SCAN          | PRODUCT_IDX  |       |       |            |          |
|     7 |    TABLE ACCESS BY INDEX ROWID| PRODUCT      |     1 |    28 |  1314   (2)| 00:00:01 |
|-    8 |   TABLE ACCESS FULL           | PRODUCT      | 1022K |   27M |  1314   (2)| 00:00:01 |
-------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------
   1 - access("O"."PRODUCT_ID"="P"."PRODUCT_ID")
   5 - filter(("O"."UNIT_PRICE"=:B1 AND "O"."QUANTITY">:B2))
   6 - access("O"."PRODUCT_ID"="P"."PRODUCT_ID")

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, & Directives – 12C
- **V$SQL_SHARED_CURSOR**
  - Can give clues to why plan changed
  - 70 columns showing mismatches /differences
  - Hard to view – query next slide

V$SQL_SHARED_CURSOR - SHARED_PROC.SQL

solarwinds

```
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;
/
```

QUERIES WITH MULTIPLE PLANS – OPTIMIZER MISMATCH

```
SELECT sql_id, count(*) plan_cnt
FROM v$sql
GROUP BY sql_id having count(*) >1
ORDER BY 2 DESC;
```

SQL_ID	PLAN_CNT
8mdz49zkajhw3	27
apkyc9bgsvw6h	16
99qa3zyarxvms	8
616m6uhpa2usu	7
1p5grz1gs7fjq	7
cusp9gtyj36y5	7
3za5u1vmpa2zj	7
2h0gb24h6zpnu	6
1gfaj4z5hn1kf	6
0v3dvmc22qnam	6
20vv6ttajyjzq	6

SELECT /*+ OPT_PARAM('_fix_control‘ 16391176:1') */
GROUP_TYPE, BUCKET_START, BUCKET_END, TM_GROUP_TYPE …

‘_fix_control’ is a hidden parameter for bugs - see v$system_fix_control or v$session_fix_control

```
Enter value for 1: 8mdz49zkajhw3

SQL_ID                          = 8mdz49zkajhw3
ADDRESS                         = 00000000E72B6128
CHILD_ADDRESS                   = 00000000DFE34D60
CHILD_NUMBER                    = 0
OPTIMIZER_MISMATCH              = Y
USE_FEEDBACK_STATS              = Y
-----------------------------------------------
SQL_ID                          = 8mdz49zkajhw3
ADDRESS                         = 00000000E72B6128
CHILD_ADDRESS                   = 00000000D8B088E8
CHILD_NUMBER                    = 1
OPTIMIZER_MISMATCH              = Y
USE_FEEDBACK_STATS              = Y
-----------------------------------------------
SQL_ID                          = 8mdz49zkajhw3
ADDRESS                         = 00000000E72B6128
CHILD_ADDRESS                   = 00000000E2208738
CHILD_NUMBER                    = 2
OPTIMIZER_MISMATCH              = Y
USE_FEEDBACK_STATS              = Y
-----------------------------------------------
…
```

QUERIES WITH MULTIPLE PLANS – BIND MISMATCH

solarwinds

SQL_ID	PLAN_CNT
8mdz49zkajhw3	27
apkyc9bgsvw6h	16

```
SELECT  count(*) child_cnt, m.position bind_number, m.max_length,
decode(m.datatype,1,'VARCHAR2',2,'NUMBER',m.datatype) AS datatype
FROM v$sql s, v$sql_bind_metadata m
WHERE s.sql_id = '&sql_id'
 AND s.child_address = m.address
group by  m.position,
m.max_length,decode(m.datatype,1,'VARCHAR2',2,'NUMBER',m.datatype)
having count(*) < &tot_child_cnt
 order by 4,1
```

CHILD_CNT	BIND_NUMBER	MAX_LENGTH	DATATYPE
7	6	22	NUMBER
7	6	32	VARCHAR2
11	9	32	VARCHAR2
3	9	128	VARCHAR2
13	12	22	NUMBER
1	12	32	VARCHAR2
11	16	22	NUMBER
3	16	32	VARCHAR2
12	17	22	NUMBER
2	17	32	VARCHAR2
8	18	22	NUMBER
6	18	32	VARCHAR2
1	22	22	NUMBER
13	22	32	VARCHAR2
7	25	22	NUMBER
7	25	32	VARCHAR2
10	26	22	NUMBER
4	26	32	VARCHAR2
9	28	32	VARCHAR2
4	28	128	VARCHAR2
1	28	2000	VARCHAR2
9	29	32	VARCHAR2
3	29	128	VARCHAR2
2	29	2000	VARCHAR2
2	30	32	VARCHAR2
5	30	128	VARCHAR2
7	30	2000	VARCHAR2
5	31	22	NUMBER
9	31	32	VARCHAR2

```
SQL> @shared_proc
Enter value for 1: apkyc9bgsvw6h

SQL_TEXT                       = INSERT INTO CONPT_4 (PLAN_HASH_VALUE, ID,
OPERATION, OPTIONS, OBJECT_NODE, OBJECT#, OBJECT_OWNER, OBJECT_NAME,
OBJECT_ALIAS, OBJECT_TYPE, OPTIMIZER, PARENT_ID, DEPTH, POSITION,
SEARCH_COLUMNS, COST, CARDINALITY, BYTES, OTHER_TAG, PARTITION_START,
PARTITION_STOP, PARTITION_ID, OTHER, DISTRIBUTION, CPU_COST, IO_COST,
TEMP_SPACE, ACCESS_PREDICATES, FILTER_PREDICATES, PROJECTION, TIME, QBLOCK_NAME,
REMARKS, TIMESTAMP) VALUES (:1 , :2 , :3 , :4 , :5 , :6 , :7 , :8 , :9 , :10 ,
:11 , :12 , :13 , :14 , :15 , :16 , :17 , :18 , :19 , :20 , :21 , :22 , :23 ,
:24 , :25 , :26 , :27 , :28 , :29 , :30 , :31 , :32 , :33 , :34 )
SQL_ID                         = apkyc9bgsvw6h
ADDRESS                        = 00000000E4697B68
CHILD_ADDRESS                  = 00000000DB67E378
CHILD_NUMBER                   = 0
BIND_MISMATCH                  = Y
REASON                         = <ChildNode> <ChildNumber>0</ChildNumber> <ID>40</ID>
<reason>Bindmismatch(8)</reason>
<size>4x4</size> <bind_position>11</bind_position>
<original_oacflg>3</original_oacflg> <original_oacdty>1</original_oacdty>
<new_oacdty>2</new_oacdty>
-----------------------------------------------
SQL_ID                         = apkyc9bgsvw6h
ADDRESS                        = 00000000E4697B68
CHILD_ADDRESS                  = 00000000DF3FA3F0
CHILD_NUMBER                   = 1
BIND_MISMATCH                  = Y
REASON                         = <ChildNode><ChildNumber>1</ChildNumber> <ID>40</ID>
<reason>Bindmismatch(8)</reason>
<size>4x4</size> <bind_position>15</bind_position>
<original_oacflg>3</original_oacflg> <original_oacdty>1</original_oacdty>
<new_oacdty>2</new_oacdty>
-----------------------------------------------
SQL_ID                         = apkyc9bgsvw6h
ADDRESS                        = 00000000E4697B68
CHILD_ADDRESS                  = 00000000DF53AEF8
CHILD_NUMBER                   = 2
BIND_MISMATCH                  = Y
REASON                         =
...
```

EXAMPLE OF ADAPTIVE PLANS

Select /* jg */ p.product_name from order_items o, product p where o.unit_price = :b1 and o.quantity > :b2 and o.product_id = p.product_id;

```
SQL> @shared_proc
Enter value for 1: 6jmbvq3nqkvak

SQL_TEXT                              = select /* jg */ p.product_name from order_items
o, product p where o.unit_price = :b1 and
o.quantity > :b2 and o.product_id =
p.product_id
SQL_ID                                = 6jmbvq3nqkvak
ADDRESS                               = 000007FF57871158
CHILD_ADDRESS                         = 000007FF28E03608
CHILD_NUMBER                          = 0
REASON                                =
<ChildNode><ChildNumber>0</ChildNumber><ID>3</ID><reason>Optimizer
mismatch(12)</reason><size>2x312</size><optimizer_adaptive_reporting_only> true
false                 </optimizer_adaptive_reporting_only></ChildNode>
---------------------------------------------------
SQL_TEXT                              = select /* jg */ p.product_name from order_items
o, product p where o.unit_price = :b1 and o.quantity > :b2 and o.product_id =
p.product_id
SQL_ID                                = 6jmbvq3nqkvak
ADDRESS                               = 000007FF57871158
CHILD_ADDRESS                         = 000007FF2642CC48
CHILD_NUMBER                          = 1
OPTIMIZER_MISMATCH                    = Y
USE_FEEDBACK_STATS                    = Y
REASON                                =
<ChildNode><ChildNumber>1</ChildNumber><ID>49</ID><reason>Auto Reoptimization
Mismatch(1)</reason><size>3x4</size><kxscflg>32</kxscflg><kxscfl4>4194560</kxscf
l4><dnum_kksfcxe>1</dnum_kksfcxe></ChildNode>
---------------------------------------------------
SQL_TEXT                              = select /* jg */ p.product_name from order_items
o, product p where o.unit_price = :b1 and o.quantity > :b2 and o.product_id =
p.product_id
SQL_ID                                = 6jmbvq3nqkvak
ADDRESS                               = 000007FF57871158
CHILD_ADDRESS                         = 000007FF28471AC0
CHILD_NUMBER                          = 2
REASON                                =
---------------------------------------------------
```

MY V$SQL_SHARED_CURSOR

solarwinds

```
BIND_EQUIV_FAILURE          = Bind value's selectivity does not match that used to optimize the existing child cursor
BIND_LENGTH_UPGRADEABLE     = Bind length(s) for current cursor are longer than bind length(s) used to build the child cursor
BIND_MISMATCH               = Bind metadata does not match existing child cursor
HASH_MATCH_FAILED           = No existing child cursors have the unsafe literal bind hash values required by the current cursor
LANGUAGE_MISMATCH           = Language handle does not match the existing child cursor
LOAD_OPTIMIZER_STATS        = Hard parse is forced in order to initialize extended cursor sharing
OPTIMIZER_MISMATCH          = Optimizer environment does not match the existing child cursor
OPTIMIZER_MODE_MISMATCH     = Parameter OPTIMIZER_MODE mismatch (for example, all_rows versus first_rows_1)
ROLL_INVALID_MISMATCH       = Marked for rolling invalidation and invalidation window exceeded
TOP_LEVEL_RPI_CURSOR        = Is top level RPI cursor
USE_FEEDBACK_STATS          = Hard parse is forced so the optimizer can reoptimize the query with improved cardinality estimates
```

```
DECLARE  c  NUMBER; col_cnt   NUMBER; col_rec   DBMS_SQL.DESC_TAB;  col_value VARCHAR2(4000); ret_val  NUMBER; v_sql_id  VARCHAR2(100);
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', 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') THEN
                                v_sql_id := col_rec(idx).col_name || ' = '|| col_value;
                           ELSIF col_value = 'Y' THEN
                                DBMS_OUTPUT.PUT_LINE(v_sql_id ||' ' ||RPAD(col_rec(idx).col_name, 30) || ' = ' || col_value);
                           END IF;
                           END LOOP;
                     END LOOP;
                     DBMS_SQL.CLOSE_CURSOR(C);
END;
/
```

```
SQL_ID = 49kx5t46pwa49 USE_FEEDBACK_STATS                = Y
SQL_ID = 3azgx3uzzsa4n USE_FEEDBACK_STATS                = Y
SQL_ID = cx4fsc3f1ca4q ROLL_INVALID_MISMATCH             = Y
SQL_ID = 01uy9sb7w8a9g HASH_MATCH_FAILED                 = Y
SQL_ID = 1mcmpjun5haf6 ROLL_INVALID_MISMATCH             = Y
SQL_ID = 6h0fx8y7hsafh USE_FEEDBACK_STATS                = Y
SQL_ID = cr8ddsaqb4aga USE_FEEDBACK_STATS                = Y
SQL_ID = 0qfk8jfqqsard ROLL_INVALID_MISMATCH             = Y
SQL_ID = 4rdmqk9nchaup USE_FEEDBACK_STATS                = Y
SQL_ID = 71s5km0f48aw5 HASH_MATCH_FAILED                 = Y
SQL_ID = 6c0q3ytfvsaw8 USE_FEEDBACK_STATS                = Y
SQL_ID = 3b5mwq5wcwawb USE_FEEDBACK_STATS                = Y
```

V$SQLPLAN & DBMS_XPLAN.DISPLAY_CURSOR

```
SELECT /*+ opt_param('parallel_execution_enabled', 'false') */
      /* EXEC_FROM_DBMS_XPLAN */
     id, parent_id, partition_id, timestamp, optimizer, position, search_columns, depth,
     operation, options, object_name, object_owner, object_type, null as object_instance,
     cardinality, bytes, temp_space, cost, io_cost, cpu_cost, time, partition_start, partition_stop,
     object_node, other_tag, distribution, null, access_predicates, filter_predicates,
     other, null, null, other_xml, sql_profile, sql_plan_baseline, null, null,  null, null, null, null,
     null,null, null, null, null, null, null, null, null, null from
     (select /*+ no_merge */  vp.id id, vp.parent_id parent_id, vp.partition_id,  vp.timestamp,
           vp.optimizer, vp.search_columns, vp.depth depth, vp.position position,
          vp.operation operation, vp.options options, vp.cost cost, vp.time time, vp.cardinality cardinality,
          vp.bytes bytes, vp.object_node object_node, vp.object_name object_name,
          vp.object_owner object_owner, vp.object_type object_type, null as object_instance,
          vp.other_tag other_tag, vp.partition_start partition_start, vp.partition_stop partition_stop,
          vp.distribution distribution, vp.temp_space temp_space, vp.io_cost io_cost,  vp.cpu_cost cpu_cost,
          vp.filter_predicates filter_predicates, vp.access_predicates access_predicates, vp.other other,
          vp.projection projection, vp.qblock_name qblock_name, vp.object_alias object_alias,
          vp.other_xml other_xml, v$sql.sql_profile sql_profile, v$sql.sql_plan_baseline sql_plan_baseline,
          0 starts, 0 outrows, 0 crgets, 0 cugets,0 reads,0 writes, 0 etime,0 mem_opt,0 mem_one,
          null last_mem_used, null last_mem_usage,0 opt_cnt,0 one_cnt,0 multi_cnt,0 max_tmp,0 last_tmp
from V$SQL_PLAN vp, v$sql
where vp.SQL_ID = &sql_id  and vp.child_number=&child
and vp.SQL_ID = v$sql.SQL_ID
and v$sql.is_obsolete = 'N'
and v$sql.address = vp.address
and v$sql.child_number= &child)
order by id;
```

select * from table(dbms_xplan.display_cursor('&sql_id', &child_number));

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  d6j5vmgrxvdfg, child number 1
-------------------------------------
select SAMPDATA.ID as SERIESID,LOOKUP.NAME as SERIES,SAMPDATA.DATEHOUR
as CATEGORYID, nvl(SAMPDATA.TOT,0) as VALUE, LOOKUP.VALUE as RANK from
(select SAMP.DATEHOUR, DATA.TOT,SAMP.ID from  (select distinct
SUMTABLE.DATEHOUR as DATEHOUR, T1.ID from CON_SAMPLE_SUM_6 SUMTABLE,
DATEHOUR, sum(SUMTABLE.TIMESECS) as TOT from CON_SQL_SUM_6 SUMTABLE,
CONDATA1_TEMP T1 where SUMTABLE.DATEHOUR between :3  and :4  and
sum(SUMTABLE.TIMESECS) as TOT from CON_SQL_SUM_6 SUMTABLE
...

Plan hash value: 535812656
------------------------------------------------------------------------------------------------------------------------
| Id  | Operation             | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |                 |       |       |    24 (100)|          |
|   1 |  SORT ORDER BY        |                 |    26 | 15158 |    24  (17)| 00:00:01 |
|*  2 |   HASH JOIN RIGHT OUTER                 |                 |    26 | 15158 |    23  (14)| 00:00:01 |
|   3 |    VIEW               |                 |     3 |   102 |    16  (13)| 00:00:01 |
|   4 |     UNION-ALL         |                 |       |       |            |          |
|   5 |      HASH GROUP BY    |                 |     1 |    39 |     9  (12)| 00:00:01 |
|*  6 |       FILTER          |                 |       |       |            |          |
|*  7 |        HASH JOIN ANTI |                 |     9 |   351 |     6   (0)| 00:00:01 |
|   8 |         TABLE ACCESS BY INDEX ROWID BATCHED  | CON_SQL_SUM_6   |    11 |   286 |     4   (0)| 00:00:01 |
|*  9 |          INDEX RANGE SCAN                    | SYS_C0012407    |    11 |       |     3   (0)| 00:00:01 |
|  10 |         TABLE ACCESS FULL                    | CONDATA1_TEMP   |     2 |    26 |     2   (0)| 00:00:01 |
|  11 |        SORT AGGREGATE |                 |     1 |       |            |          |
|  12 |         TABLE ACCESS FULL                    | CONDATA1_TEMP   |     2 |       |     2   (0)| 00:00:01 |
...
|* 30 |        INDEX RANGE SCAN                      | SYS_C0012423    |    13 |   130 |     1   (0)| 00:00:01 |
------------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("SAMP"."DATEHOUR"="DATA"."DATEHOUR" AND "SAMP"."ID"="DATA"."ID")
   6 - filter((:4>=:3 AND =500))
...
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - statistics feedback used for this statement
   - 6 Sql Plan Directives used for this statement
```

V$SQLPLAN & DBMS_XPLAN.DISPLAY_CURSOR – CONT.

solarwinds

OTHER_XML column contains notes, peeked binds, hints, & display map information

```
SELECT /*+ opt_param('parallel_execution_enabled', 'false') */
extractvalue(xmlval, '/*/info[@type = "sql_profile"]') sql_profile
, extractvalue(xmlval, '/*/info[@type = "sql_patch"]') sql_patch
, extractvalue(xmlval, '/*/info[@type = "baseline"]') baseline
, extractvalue(xmlval, '/*/info[@type = "outline"]') outline
, extractvalue(xmlval, '/*/info[@type = "dynamic_sampling"]') dynamic_sampling
, extractvalue(xmlval, '/*/info[@type = "dop"]') dop
, extractvalue(xmlval, '/*/info[@type = "dop_reason"]') dop_reason
, extractvalue(xmlval, '/*/info[@type = "pdml_reason"]') pdml_reason
, extractvalue(xmlval, '/*/info[@type = "idl_reason"]') idl_reason
, extractvalue(xmlval, '/*/info[@type = "queuing_reason"]') queuing_reason
, extractvalue(xmlval, '/*/info[@type = "px_in_memory"]') px_in_memory
, extractvalue(xmlval, '/*/info[@type = "px_in_memory_imc"]') px_in_memory_imc
, extractvalue(xmlval, '/*/info[@type = "row_shipping"]') row_shipping
, extractvalue(xmlval, '/*/info[@type = "index_size"]') index_size
, extractvalue(xmlval, '/*/info[@type = "result_checksum"]') result_checksum
, extractvalue(xmlval, '/*/info[@type = "cardinality_feedback"]') cardinality_feedback
, extractvalue(xmlval, '/*/info[@type = "performance_feedback"]') performance_feedback
, extractvalue(xmlval, '/*/info[@type = "xml_suboptimal"]') xml_suboptimal
, extractvalue(xmlval, '/*/info[@type = "adaptive_plan"]') adaptive_plan
, extractvalue(xmlval, '/*/spd/cu') spd_cu
, extractvalue(xmlval, '/*/info[@type = "gtt_session_st"]') gtt_session_st
, extractvalue(xmlval,'/*/info[@type = "plan_hash"]') plan_hash
from
(select xmltype(replace(other_xml,"","")) xmlval from v$sql_plan
where sql_id = '&sqlid' and child_number=&child and other_xml is not null);
```

```
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - statistics feedback used for this statement
   - 6 Sql Plan Directives used for this statement
```

SQL_PROFIL	SQL_PATCH	BASELINE	OUTLINE	DYNAMIC_SA	DOP	DOP_REASON	PDML_REASO	IDL_REASON	QUEUING_RE	PX_IN_MEMO
PX_IN_MEMO	ROW_SHIPPI	INDEX_SIZE	RESULT_CHE	CARDINALIT	PERFORMANC	XML_SUBOPT	ADAPTIVE_P	SPD_CU	GTT_SESSIO	PLAN_HASH
				2						
				yes				6		535812656

WHICH FEATURES MAY IMPACT THE PLAN

solarwinds

Bind Variable Peeking / Adaptive Cursor Sharing Example

```
c:\ORACLE\diag\rdbms\cece\trace> tkprof cece_ora_7264.trc f40_x5.lst explain=scott/scott


BEGIN :P1 :='40'; END;

********************************************************************************

SELECT e.empno EID, e.ename "Employee_name",  d.dname "Department", e.hiredate "Date_Hired"
FROM  emp e,  dept d WHERE  d.deptno =  :P1 AND e.deptno = d.deptno

call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch      265      0.01       0.00          0        566          0        3958
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total      267      0.01       0.00          0        566          0        3958

Optimizer mode: ALL_ROWS

Rows     Row Source Operation
-------  ---------------------------------------------------
   3958  NESTED LOOPS  (cr=566 pr=0 pw=0 time=0 us cost=4 size=2772 card=77)
      1   TABLE ACCESS BY INDEX ROWID DEPT (cr=3 pr=0 pw=0 time=0 us cost=2 size=11 card=1)
      1    INDEX UNIQUE SCAN PK_DEPT (cr=2 pr=0 pw=0 time=0 us cost=1 size=0 card=1)(object id 69947)
   3958   TABLE ACCESS BY INDEX ROWID EMP (cr=563 pr=0 pw=0 time=0 us cost=2 size=1925 card=77)
   3958    INDEX RANGE SCAN EMP_DEPTNO (cr=273 pr=0 pw=0 time=0 us cost=1 size=0 card=77)(object id 183864)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   MODE: ALL_ROWS
   3958   NESTED LOOPS
      1    TABLE ACCESS   MODE: ANALYZED (BY INDEX ROWID) OF 'DEPT'
               (TABLE)
      1     INDEX   MODE: ANALYZED (UNIQUE SCAN) OF 'PK_DEPT' (INDEX
               (UNIQUE))
   3958    TABLE ACCESS   MODE: ANALYZED (FULL) OF 'EMP' (TABLE)
```

DEPTNO	COUNT(*)
10	77
20	1500
30	478
40	3958

First ran with :P1=10

V$SQL - IS_BIND_SENSITIVE: optimizer peeked –plan may change
V$SQL - IS_BIND_AWARE: 'Y' after query has been marked bind sensitive
New Views: V$SQL_CS_HISTOGRAM
V$SQL_CS_SELECTIVITY
V$SQL_CS_STATISTICS

BIND VARIABLE PEEKING / ADAPTIVE CURSOR SHARING

solarwinds

1st run with :p1=10

```
SQL>  select * from table(dbms_xplan.display_cursor('29g1b5n68kyns',null,'+peeked_binds'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
SQL_ID  29g1b5n68kyns, child number 0
-------------------------------------
select e.empno EOD, e.ename "Employee_name", d.dname "Department",
e.hiredate "Date_Hired" from emp e, dept d where d.deptno = :p1 and
e.deptno = d.deptno

Plan hash value: 2189666631
------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |             |       |       |     6 (100)|          |
|   1 |  MERGE JOIN CARTESIAN                 |             |    78 |  2652 |     6   (0)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL                   | DEPT        |     1 |    13 |     3   (0)| 00:00:01 |
|   3 |   BUFFER SORT                         |             |    78 |  1638 |     3   (0)| 00:00:01 |
|   4 |    TABLE ACCESS BY INDEX ROWID BATCHED| EMP         |    78 |  1638 |     3   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN                  | EMP_DEPTNO  |    78 |       |     2   (0)| 00:00:01 |
------------------------------------------------------------------------------------------------------
Peeked Binds (identified by position):
--------------------------------------
   1 - :P1 (NUMBER): 10

Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter("D"."DEPTNO"=:P1)
   5 - access("E"."DEPTNO"=:P1)
```

Changed after 2nd execution of :p1=40

```
SQL>   select sql_id, substr(sql_text,1,30) text, child_number, plan_hash_value,
  2         IS_BIND_SENSITIVE,IS_BIND_AWARE,BIND_DATA  from v$sql
  3* where sql_text like 'select e.empno EOD, e.ename%';

SQL_ID        TEXT                           CHILD_NUMBER PLAN_HASH_VALUE I I
------------- ------------------------------ ------------ --------------- - -
BIND_DATA
------------------------------------------------------------------------------------------------------
29g1b5n68kyns select e.empno EOD, e.ename "E            0      2189666631 Y N
BEDA0C100200583C989D000101C0021602C10B

29g1b5n68kyns select e.empno EOD, e.ename "E            1       615168685 Y Y
BEDA0C100200583C9938000101C0021602C129

SQL>  select * from table(dbms_xplan.display_cursor('29g1b5n68kyns',1,'+peeked_binds'));

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------
SQL_ID  29g1b5n68kyns, child number 1
-------------------------------------
select e.empno EOD, e.ename "Employee_name", d.dname "Department",
e.hiredate "Date_Hired" from emp e, dept d where d.deptno = :p1 and
e.deptno = d.deptno

Plan hash value: 615168685
-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |   623 (100)|          |
|*  1 |  HASH JOIN         |      |  368K|    11M|   623   (1)| 00:00:01 |
|*  2 |   TABLE ACCESS FULL| DEPT |     1 |    13 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| EMP  |   368K|  7563K|   619   (1)| 00:00:01 |
-----------------------------------------------------------------------------
Peeked Binds (identified by position):
--------------------------------------
   1 - :P1 (NUMBER): 40
```

BIND VARIABLE PEEKING / ADAPTIVE CURSOR SHARING

solarwinds

	SQL_ID	TEXT	CHILD_NUMBER	PLAN_HASH_VALUE	I	I
:P1	BIND_DATA					
10	29g1b5n68kyns	select e.empno EOD, e.ename "E	0	2189666631	Y	N
	BEDA0C100200583C989D000101C0021602C10B					
40	29g1b5n68kyns	select e.empno EOD, e.ename "E	1	615168685	Y	Y
	BEDA0C100200583CA1E3000101C0021602C129					
10	29g1b5n68kyns	select e.empno EOD, e.ename "E	2	2189666631	Y	Y
	BEDA0C100200583C9D6D000101C0021602C10B					
20	29g1b5n68kyns	select e.empno EOD, e.ename "E	3	2189666631	Y	Y
	BEDA0C100200583CA16A000101C0021602C115					
30	29g1b5n68kyns	select e.empno EOD, e.ename "E	4	2189666631	Y	Y
	BEDA0C100200583CA1B2000101C0021602C11F					

```
SELECT * FROM TABLE
(DBMS_SQLTUNE.EXTRACT_BINDS
('BEDA0C100200583C989D000101C0021602C10B'));
```

DATATYPE_STRING	MAX_LENGTH	VALUE_STRING
NUMBER	22	10

DYNAMIC STATISTICS (DYNAMIC SAMPLING IN 9.2)

- Missing Statistics
- Stale Statistics
- Insufficient Statistics
- Parallel Execution
- SQL Plan Directives

```
SQL> EXPLAIN PLAN FOR
2  SELECT product_id, product_name, product_description
3  FROM product_information
4  WHERE category_id IN (12,126)
5  AND product_name LIKE '%j%';

Explained.

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

-------------------------------------------------------------------------------
Plan hash value: 2715330242
-------------------------------------------------------------------------------
| Id  | Operation         | Name                | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                     | 65537 |  8064K|  9169   (1)| 00:00:01
|*  1 |  TABLE ACCESS FULL| PRODUCT_INFORMATION | 65537 |  8064K|  9169   (1)| 00:00:01
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
1 - filter("PRODUCT_NAME" LIKE '%j%' A
        "CATEGORY_ID"=126) AND "PRODUC
```

Estimates over 6X off!

```
SQL> select count(*)
2  from product_information
3  where category_id in (12,126)
4  and product_name like '%j%';

COUNT(*)
----------
393221
```

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

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

Session altered.

SQL> EXPLAIN PLAN FOR
2  SELECT product_id, product_name, product_description
3  FROM product_information
4  WHERE category_id IN (12,126)
5  AND product_name LIKE '%j%';

Explained.

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

-------------------------------------------------------------------------------
Plan hash value: 2715330242
-------------------------------------------------------------------------------
| Id  | Operation         | Name                | Rows  | Bytes | Cost (%CPU)| Time
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |                     |  349K|    42M|  9169   (1)| 00:00:01
|*  1 |  TABLE ACCESS FULL| PRODUCT_INFORMATION |  349K|    42M|  9169   (1)| 00:00:01
-------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
1 - filter("PRODUCT_NAME" LIKE '%j%' AND ("CATEGORY_ID"=12 OR
        "CATEGORY_ID"=126) AND "PRODUCT_NAME" IS NOT NULL)
Note
-----
- dynamic statistics used: dynamic sampling (level=AUTO)
```

STATISTICS FEEDBACK

Estimated cardinalities can be incorrect for many reasons

- **missing statistics, inaccurate statistics, or complex predicates**

After 1st execution, estimates are compared with actual rows

- **If they differ significantly, optimizer stores correct estimates for future use**
  - Statistics feedback = OPT_ESTIMATE hints in V$SQL_REOPTIMIZATION_HINTS

```
SQL> select s.sql_id,substr(sql_text,1,60) sql_text,s.child_number,hint_text
  2  from v$sql s, V$SQL_REOPTIMIZATION_HINTS h
  3  where s.sql_id = h.sql_id
  4* and s.child_number = h.child_number;

SQL_ID        SQL_TEXT
------------- ------------------------------------------------------------
CHILD_NUMBER
------------
HINT_TEXT
--------------------------------------------------------------------------------
6x007v5whw055 select SAMPDATA.ID as SERIESID,LOOKUP.NAME as SERIES,SAMPDAT
            0
OPT_ESTIMATE (@"SEL$F5BB74E1" JOIN ("DATA"@"SEL$2" "LOOKUP"@"SEL$1" "SAMP"@"SEL$
2") ROWS=14.000000 )
```

Can create a SQL PLAN DIRECTIVE for other SQL statements

- **They benefit from information obtained during initial execution**

STATISTICS FEEDBACK

select * from table(DBMS_XPLAN.DISPLAY_CURSOR(FORMAT=>'ALLSTATS LAST'));

```
SQL_ID  funnss95hcebs, 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 (c.cust_first_name like '%jig%' 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
'%jbtqiwxr5E%')

Plan hash value: 2318532850

-----------------------------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                           | Name             | Starts | E-Rows | A-Rows |   A-Time   | Buffers | Reads  | OMem |  1Mem | Used-Mem |
-----------------------------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                  |      1 |        |      5 |00:01:10.11 |   190K|    145K|      |       |          |
|*  1 |  HASH JOIN                          |                  |      1 |      5 |      5 |00:01:10.11 |   190K|    145K| 3483K|  1239K| 4462K (0)|
|*  2 |   HASH JOIN                         |                  |      1 |      3 |  26077 |00:01:09.00 |   180K|    140K| 2324K|  1378K| 2781K (0)|
|   3 |    MERGE JOIN CARTESIAN             |                  |      1 |      1 |  18821 |00:00:38.86 |   121K|  81694 |      |       |          |
|   4 |     NESTED LOOPS                    |                  |      1 |        |  18821 |00:00:37.76 |    98165 |  76710 |      |       |          |
|   5 |      NESTED LOOPS                   |                  |      1 |      1 |  18821 |00:00:22.27 |    79341 |  67659 |      |       |          |
|*  6 |       TABLE ACCESS FULL             | CUSTOMERS        |      1 |   9440 |   9440 |00:00:06.77 |    60395 |  60370 |      |       |          |
|*  7 |       INDEX RANGE SCAN              | ORD_CUSTOMER_IX  |   9440 |      2 |  18821 |00:00:27.55 |    18946 |   7289 |      |       |          |
|   8 |      TABLE ACCESS BY INDEX ROWID    | ORDERS           |  18821 |      1 |  18821 |00:00:22.90 |    18824 |   9051 |      |       |          |
|   9 |     BUFFER SORT                     |                  |  18821 |      1 |  18821 |00:00:01.04 |    23225 |   4984 |  2048 |  2048 | 2048  (0)|
|  10 |      VIEW                           | VW_NSO_1         |      1 |      1 |      1 |00:00:00.96 |    23225 |   4984 |      |       |          |
|  11 |       HASH UNIQUE                   |                  |      1 |      1 |      1 |00:00:00.96 |    23225 |   4984 |  2294K|  2294K|  435K (0)|
|* 12 |        HASH JOIN SEMI               |                  |      1 |      1 |      1 |00:00:00.96 |    23225 |   4984 |  2293K|  2293K|  429K (0)|
|* 13 |         TABLE ACCESS FULL           | PRODUCT          |      1 |      1 |      1 |00:00:00.43 |     9849 |   4984 |      |       |          |
|* 14 |         TABLE ACCESS FULL           | INVENTORIES      |      1 |   1257K|   1088K|00:00:00.50 |    13376 |      0 |      |       |          |
|  15 |    TABLE ACCESS FULL                | ORDER_ITEMS      |      1 |      3 |    11M|00:00:05.41 |    59010 |  58760 |      |       |          |
|  16 |   TABLE ACCESS FULL                 | PRODUCT          |      1 |   1172K|   1311K|00:00:00.32 |     9850 |   4984 |      |       |          |
-----------------------------------------------------------------------------------------------------------------------------------------------------

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

   1 - access("I"."PRODUCT_ID"="P"."PRODUCT_ID" AND "P"."PRODUCT_ID"="PRODUCT_ID")
   2 - access("O"."ORDER_ID"="I"."ORDER_ID")
   6 - filter(("C"."CUST_FIRST_NAME" LIKE '%jig%' OR "C"."CUST_LAST_NAME" LIKE '%gri%'))
   7 - access("C"."CUSTOMER_ID"="O"."CUSTOMER_ID")
  12 - access("I"."PRODUCT_ID"="P"."PRODUCT_ID")
  13 - filter(("PRODUCT_NAME" IS NOT NULL AND "PRODUCT_NAME" LIKE '%jbtqiwxr5E%'))
  14 - filter("I"."QUANTITY_ON_HAND">10)

Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - statistics feedback used for this statement
   - this is an adaptive plan
```

STATISTICS FEEDBACK – ANOTHER EXAMPLE

```
SQL> select s.sql_id,substr(sql_text,1,60) sql_text,s.child_number,hint_text
  2  from v$sql s, V$SQL_REOPTIMIZATION_HINTS h
  3  where s.sql_id = h.sql_id
  4  and s.child_number = h.child_number
  5* and s.sql_id = '255s88y9tq1nf'

SQL_ID        SQL_TEXT                                                    CHILD_NUMBER
------------- ------------------------------------------------------------ ------------
HINT_TEXT
--------------------------------------------------------------------------------
255s88y9tq1nf  SELECT sum(SW.QP)/100.0 FROM CONSW_4 SW, CONEV_4 EV where S            0
OPT_ESTIMATE (@"SEL$1" JOIN ("SW"@"SEL$1" "EV"@"SEL$1") ROWS=1.000000 )


SQL> SELECT sum(SW.QP)/100.0
  2  FROM CONSW_4 SW, CONEV_4 EV
  3  where SW.KEEQ=EV.ID and lower(EV.NAME)='direct path read' and SW.D > :p1;

SUM(SW.QP)/100.0
----------------
             956
```

```
SQL> select * from table (dbms_xplan.display_cursor('&sql','&child',format=>'+adaptive'));
Enter value for sql: 255s88y9tq1nf
Enter value for child: 0

Plan hash value: 689912439
-------------------------------------------------------------------------------------------------------------------
|   Id  | Operation                           | Name        | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------------
|     0 | SELECT STATEMENT                    |             |       |       |  5776 (100)|          |
|     1 |  SORT AGGREGATE                     |             |     1 |    44 |            |          |
|- *  2 |   HASH JOIN                         |             |   977 | 42988 |  5776   (1)| 00:00:01 |
|     3 |    NESTED LOOPS                     |             |   977 | 42988 |  5776   (1)| 00:00:01 |
|     4 |     NESTED LOOPS                    |             | 95024 | 42988 |  5776   (1)| 00:00:01 |
|-    5 |      STATISTICS COLLECTOR           |             |       |       |            |          |
|  *  6 |       TABLE ACCESS FULL             | CONEV_4     |     1 |    29 |     7   (0)| 00:00:01 |
|  *  7 |      INDEX RANGE SCAN               | IX1_CONSW_4 | 95024 |       |   232   (1)| 00:00:01 |
|  *  8 |     TABLE ACCESS BY INDEX ROWID     | CONSW_4     |   977 | 14655 |  5769   (1)| 00:00:01 |
|-    9 |    TABLE ACCESS BY INDEX ROWID BATCHED| CONSW_4   |   977 | 14655 |  5769   (1)| 00:00:01 |
|- * 10 |     INDEX RANGE SCAN                | IX1_CONSW_4 | 95024 |       |   232   (1)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - access("SW"."KEEQ"="EV"."ID")
   6 - filter(LOWER("EV"."NAME")='direct path read')
   etc...
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
   - this is an adaptive plan (rows marked '-' are inactive)
   - 3 Sql Plan Directives used for this statement
```

SQL PLAN DIRECTIVES

solarwinds

Are additional Instructions for missing column group statistics or histograms

- **Dynamic sampling performed on directive**
  - Until statistics are gathered for the column group (e.g. City / State / Country)

Not tied to a specific sql statement – defined on a query expression

- **Can be used by similar queries**

Are created in shared_pool & periodically written to SYSAUX tablespace

- **DBA_SQL_PLAN_DIRECTIVES**
- **DBA_SQL_PLAN_DIR_OBJECTS**

Use DBMS_STATS extended functions & procedures

- **CREATE_EXTENDED_STATS**
- **SHOW_EXTENDED_STATS_NAME**
- **DROP_EXENTED_STATS**

SQL PLAN DIRECTIVES – CONT.

```
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 ('DPA')
ORDER BY 1,2,3,4,5;
```

SELECT sum(SW.QP)/100.0 FROM CONSW_4 SW, CONEV_4 EV where SW.KEEQ=EV.ID and lower(EV.NAME)='direct path read' and SW.D > :p1;

```
DIR_ID                              OWNER           OBJECT_NAME         COL_NAME            OBJECT TYPE
----------------------------------- --------------- ------------------- ------------------- ------ ---------------
STATE      REASON
---------- -----------------------------------
10597117826578484594                DPA             CONEV_4                                 TABLE  DYNAMIC_SAMPLING
USABLE     JOIN CARDINALITY MISESTIMATE

10597117826578484594                DPA             CONSW_4                                 TABLE  DYNAMIC_SAMPLING
USABLE     JOIN CARDINALITY MISESTIMATE

235444056559041457                  DPA             CONSW_4             D                   COLUMN DYNAMIC_SAMPLING
USABLE     SINGLE TABLE CARDINALITY MISESTIMATE

7299144730421644734                 DPA             CONEV_4             NAME                COLUMN DYNAMIC_SAMPLING
USABLE     SINGLE TABLE CARDINALITY MISESTIMATE

etc...

31 rows selected.
```

HOW TO MANAGE SQL PLAN DIRECTIVES

solarwinds

Use DBMS_SPD package to manage

- **Can’t manually create directives**

Name	P / F	Task
ALTER_SQL_PLAN_DIRECTIVE	P	Changes either STATE or AUTO_DROP
DROP_SQL_PLAN_DIRECTIVE	P	Drops SQL Plan Directive (SPD)
FLUSH_SQL_PLAN_DIRECTIVE	P	Flushes SQL Plan Directives out of SGA
CREATE_STGTAB_DIRECTIVE	P	Staging table for exporting SPDs
PACK_STGTAB_DIRECTIVE	F	Exports SPDs in to Staging table
UNPACK_STGTAB_DIRECTIVE	F	Imports SPDs from Staging table
GET_PREFS	F	Get setting for SPD_RETENTION_WEEKS
SET_PREFS	P	Sets SPD_RETENTION_WEEKS (default set to 53 weeks)
TRANSFER_SPD_FOR_DP	P	Undocumented

DON’T WANT ANY DIRECTIVES

Drop all existing SPD

Alter system / session set “_optimizer_dsdir_usage_control”=0;

- **Not Recommended**

Tip: Monitor SQL DIRECTIVES and their states closely

- **Create extended statistics**

```
SQL>  select dbms_stats.create_extended_stats('soe','employee','(work_city,work_state,work_zip)') from dual;


DBMS_STATS.CREATE_EXTENDED_STATS('SOE','EMPLOYEE','(WORK_CITY,WORK_STATE,WORK_ZIP)')
--------------------------------------------------------------------------------
SYS_STUP9CR43YUD9SD92EXG_EYWYK
```

- **Remove directives when supported by valid statistics.**

12.2 OPTIMIZER CHANGES

solarwinds

New init.ora parameters

- **OPTIMIZER_ADAPTIVE_PLANS (Default = TRUE)**
  - Adaptive joins
  - Bitmap pruning
  - Parallel distribution method

- **OPTIMIZER_ADAPTIVE_STATISTICS (Default = False)**
  - SQL Plan Directives (SPDs) for query optimization
  - Statistics feedback (for joins only)
  - Adaptive dynamic sampling for parallel queries
  - Performance feedback

Obsolete

- **OPTIMIZER_ADAPTIVE_FEATURES**

12.2 OPTIMIZER CHANGES

solarwinds

SQL Plan Directives

- **Control of Auto Creation of Column Group Statistics**
- **New DBMS_STATS preference**
  - AUTO_STAT_EXTENSIONS (DEFAULT=OFF)
  - EXEC DBMS_STATS.SET_GLOBAL_PREFS('AUTO_STAT_EXTENSIONS','ON')
- **Can view in DBA_STAT_EXTENSIONS**

```
select owner,
        table_name,
        extension,
        extension_name
from dba_stat_extensions
where creator = 'SYSTEM'
order by owner,table_name,extension_name;
```

DBMS_SPD

solarwinds

Example of dropping all directives for ‘SOE’ user

```
EXEC DBMS_SPD. FLUSH_SQL_PLAN_DIRECTIVE;

BEGIN

FOR get_rec in (SELECT distinct TO_CHAR(d.directive_id) dir_id
            FROM dba_sql_plan_directives d, dba_sql_plan_dir_objects o
            WHERE d.directive_id = o.directive_id
            AND o.owner in ('SOE')) LOOP
DBMS_SPD.DROP_SQL_PLAN_DIRECTIVE(get_rec.dir_id);
end loop;
end;
/

commit;
```

HOW TO STABILIZE PLANS

Baselines

Patches

Profiles

SQL PLAN MANAGEMENT (BASELINES)

solarwinds

11g feature helps stop performance regression via plan changes

- **Uses baselines to guarantee only better plans are used**

12c 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 in 12c**

```
var task varchar2(1000);
var evolve varchar2(100);
var imple varchar2(1000);
var rpt clob;
exec :task := DBMS_SPM.CREATE_EVOLVE_TASK(sql_handle=>'&sql_handle');
exec :evolve := DBMS_SPM.EXECUTE_EVOLVE_TASK(task_name=>:task);
exec :imple :=DBMS_SPM.IMPLEMENT_EVOLVE_TASK(task_name=>:task,FORCE=>true);
exec :rpt := DBMS_SPM.REPORT_EVOLVE_TASK(task_name=>:task,type=>'TEXT', execution_name=>:evolve);
print
```

SQL PLAN MANAGEMENT (BASELINES)

solarwinds

```
------------------------------------------------------------------------------
GENERAL INFORMATION SECTION
------------------------------------------------------------------------------
 Task Information:
 ---------------------------------------------
 Task Name             : TASK_231
 Task Owner            : SOE
 Execution Name        : EXEC_241
 Execution Type        : SPM EVOLVE
 Scope                 : COMPREHENSIVE
 Status                : COMPLETED
 Started               : 01/16/2014 15:36:22
 Finished              : 01/16/2014 15:36:22
 Last Updated          : 01/16/2014 15:36:22
 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  : 0
  Number of errors           : 0
------------------------------------------------------------------------------
DETAILS SECTION
------------------------------------------------------------------------------
 Object ID           : 2
 Test Plan Name      : SQL_PLAN_9f0vf3un13snc19ac244c
 Base Plan Name      : SQL_PLAN_9f0vf3un13snc618516c2
 SQL Handle          : SQL_97036e1ea811e28c
 Parsing Schema      : SOE
 Test Plan Creator   : SOE
 SQL Text            : SELECT PRODUCTS.PRODUCT_ID, PRODUCT_NAME,
                       PRODUCT_DESCRIPTION, CATEGORY_ID, WEIGHT_CLASS,
                       WARRANTY_PERIOD, SUPPLIER_ID, PRODUCT_STATUS, LIST_PRICE,
                       MIN_PRICE, CATALOG_URL, QUANTITY_ON_HAND FROM PRODUCTS,
                       INVENTORIES WHERE PRODUCTS.PRODUCT_ID = :B2 AND
                       INVENTORIES.PRODUCT_ID = PRODUCTS.PRODUCT_ID AND ROWNUM < :B1

Bind Variables:
---------------------------
 1  -  (NUMBER):  266
 2  -  (NUMBER):  15
```

```
Execution Statistics:
-----------------------------
                    Base Plan                       Test Plan
                    ----------------------------    ----------------------------
 Elapsed Time (s):  .001386                         .001356
 CPU Time (s):      .001387                         .001387
 Buffer Gets:       286                             286
 Optimizer Cost:    5306                            5306
 Disk Reads:        0                               0
 Direct Writes:     0                               0
 Rows Processed:    1                               1
 Executions:        10                              10


FINDINGS SECTION
------------------------------------------------------------------------------
Findings (1):
-----------------------------
 1. The plan was verified in 0.29600 seconds. It failed the benefit criterion
    because its verified performance was 1.00000 times worse than that of the
    baseline plan.

------------------------------------------------------------------------------
EXPLAIN PLANS SECTION
------------------------------------------------------------------------------
```

EXAMPLE – PRODUCT QUERY

```
SELECT products.product_id, product_name, product_description, category_id, weight_class,
        warranty_period, supplier_id, product_status,  list_price, min_price, catalog_url, quantity_on_hand
FROM products, inventories
WHERE products.category_id = :b3
AND inventories.product_id = products.product_id
AND inventories.warehouse_id = :b2
AND rownum < :b1
```

SELECT sql_handle, plan_name, sql_text, enabled, accepted, fixed from dba_sql_plan_baselines;

SQL_HANDLE	PLAN_NAME	SQL_TEXT	ENA	ACC	FIX	OPT_COST	LAST_EXECUTED
SYS_SQL_fdf0214a24814e13	SYS_SQL_PLAN_24814e132c1c9d7b	SELECT PRODUCTS.PROD	YES	YES	NO	940	09-sep-11 20:21
SYS_SQL_fdf0214a24814e13	SYS_SQL_PLAN_24814e134d8f3521	SELECT PRODUCTS.PROD	YES	NO	NO	18	
SYS_SQL_fdf0214a24814e13	SYS_SQL_PLAN_24814e1357bbcbf2	SELECT PRODUCTS.PROD	YES	NO	NO	25	

```
SET SERVEROUTPUT ON LONG 10000
DECLARE rpt clob;
BEGIN
   rpt := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(
      sql_handle => 'SYS_SQL_fdf0214a24814e13');
   DBMS_OUTPUT.PUT_LINE(rpt);
END;
```

EXAMPLE – BASELINE / EVOLVE

```
SQL_HANDLE                PLAN_NAME                       SQL_TEXT                ENA ACC FIX OPT_COST LAST_EXECUTED
------------------------- ------------------------------- ----------------------- --- --- --- -------- ---------------
SYS_SQL_fdf0214a24814e13   SYS_SQL_PLAN_24814e132c1c9d7b   SELECT PRODUCTS.PROD   YES YES NO       940 09-sep-11 20:21
SYS_SQL_fdf0214a24814e13   SYS_SQL_PLAN_24814e134d8f3521   SELECT PRODUCTS.PROD   YES NO  NO        18
SYS_SQL_fdf0214a24814e13   SYS_SQL_PLAN_24814e1357bbcbf2   SELECT PRODUCTS.PROD   YES NO  NO        25
```

```
                          Evolve SQL Plan Baseline Report
--------------------------------------------------------------------------------
Inputs:
-------

SQL_HANDLE = SYS_SQL_fdf0214a24814e13
  PLAN_NAME  =
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = NO

Plan: SYS_SQL_PLAN_24814e134d8f3521
-----------------------------------
  Plan was verified: Time used .031 seconds.
  Failed performance criterion: Compound improvement ratio < 1

                     Baseline Plan      Test Plan      Improv. Ratio
                     -------------      ---------      -------------
  Execution Status:       COMPLETE       COMPLETE
  Rows Processed:                5              5
  Elapsed Time(ms):              5              0
  CPU Time(ms):                 15              0
  Buffer Gets:                1037           1037                 1
  Disk Reads:                    0              0
  Direct Writes:                 0              0
  Fetches:                       0              0
  Executions:                    1              1

Plan:
SYS_SQL_PLAN_24814e1357bbcbf2
-----------------------------
  Plan was verified: Time used .047 seconds.
  Failed performance criterion: Compound improvement ratio < 1

                     Baseline Plan      Test Plan      Improv. Ratio
                     -------------      ---------      -------------
  Execution Status:       COMPLETE       COMPLETE
  Rows Processed:                3              3
  Elapsed Time(ms):              0              0
  CPU
  Time(ms):                     15              0
  Buffer Gets:                1035           1035                 1
  Disk Reads:                    0              0
  Direct Writes:                 0              0
  Fetches:                       0              0
  Executions:                    1              1
--------------------------------------------------------------------------------
                              Report Summary
--------------------------------------------------------------------------------
Number of SQL plan baselines verified: 2.
Number of SQL plan baselines evolved: 0.
```

EXAMPLE – HIGH COST

solarwinds

```
SQL_HANDLE                     PLAN_NAME                          SQL_TEXT                ENA ACC FIX OPT_COST LAST_EXECUTED
-----------------------------  ---------------------------------  ----------------------  --- --- --- -------- ---------------
SYS_SQL_fdf0214a24814e13        SYS_SQL_PLAN_24814e132c1c9d7b      SELECT PRODUCTS.PROD    YES YES NO       940 09-sep-11 20:21
SYS_SQL_fdf0214a24814e13        SYS_SQL_PLAN_24814e134d8f3521      SELECT PRODUCTS.PROD    YES NO  NO        18
SYS_SQL_fdf0214a24814e13        SYS_SQL_PLAN_24814e1357bbcbf2      SELECT PRODUCTS.PROD    YES NO  NO        25
```

```
select * from table(dbms_xplan.display_sql_plan_baseline(sql_handle=>'SYS_SQL_fdf0214a24814e13',plan_name=>'SYS_SQL_PLAN_24814e132c1c9d7b'))
```

```
Plan name: SYS_SQL_PLAN_24814e132c1c9d7b
Enabled: YES     Fixed: NO      Accepted: YES     Origin: AUTO-CAPTURE
--------------------------------------------------------------------------------

Plan hash value: 750880835

-------------------------------------------------------------------------------------------------
| Id  | Operation                      | Name                  | Rows  | Bytes | Cost (%CPU)
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT               |                       |   107 |   340K|    29   (4)
|*  1 |  COUNT STOPKEY                 |                       |       |       |
|*  2 |   HASH JOIN                    |                       |   107 |   340K|    29   (4)
|*  3 |    HASH JOIN OUTER             |                       |    49 |   154K|    24   (5)
|   4 |     TABLE ACCESS BY INDEX ROWID| PRODUCT_INFORMATION   |    49 | 56497 |     5   (0)
|*  5 |      INDEX RANGE SCAN          | PROD_CATEGORY_IX      |    20 |       |     1   (0)
|*  6 |     TABLE ACCESS FULL          | PRODUCT_DESCRIPTIONS  |    49 |    99K|    18   (0)
|   7 |    TABLE ACCESS BY INDEX ROWID | INVENTORIES           |  8982 |   342K|     5   (0)
|*  8 |     INDEX RANGE SCAN           | INVENTORIES_IX1       |  3593 |       |     1   (0)
-------------------------------------------------------------------------------------------------


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

   1 - filter(ROWNUM<TO_NUMBER(:B1))
   2 - access("INVENTORIES"."PRODUCT_ID"="I"."PRODUCT_ID")
   3 - access("D"."PRODUCT_ID"(+)="I"."PRODUCT_ID")
   5 - access("I"."CATEGORY_ID"=TO_NUMBER(:B3))
   6 - filter("D"."LANGUAGE_ID"(+)=SYS_CONTEXT('USERENV','LANG'))
   8 - access("INVENTORIES"."WAREHOUSE_ID"=TO_NUMBER(:B2))
```

EXAMPLE – FORCE PLAN

Force Evolution of new baseline

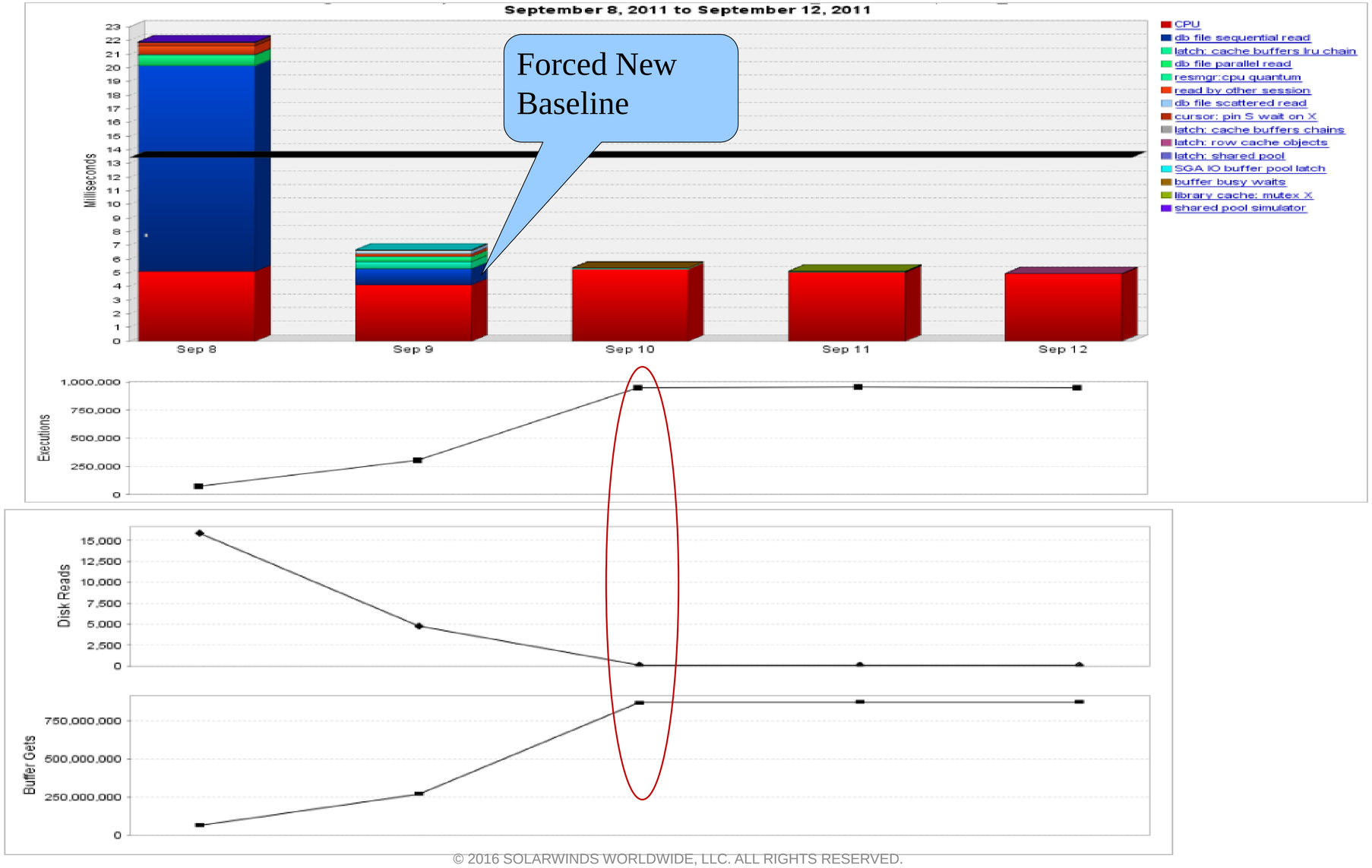
```
var task varchar2(1000);
var evolve varchar2(100);
var imple varchar2(1000);
var rpt clob;
exec :task := DBMS_SPM.CREATE_EVOLVE_TASK(sql_handle=>'&sql_handle');
exec :evolve := DBMS_SPM.EXECUTE_EVOLVE_TASK(task_name=>:task);
exec :imple :=DBMS_SPM.IMPLEMENT_EVOLVE_TASK(task_name=>:task,FORCE=>true);
exec :rpt := DBMS_SPM.REPORT_EVOLVE_TASK(task_name=>:task,type=>'TEXT', execution_name=>:evolve);
print
```

SQL_HANDLE	PLAN_NAME	SQL_TEXT	ENA	ACC	FIX	OPTIMIZER_COST
SYS_SQL_fdf0214a24814e13	SYS_SQL_PLAN_24814e132c1c9d7b	SELECT PRODUCTS.PROD	NO	YES	NO	940
SYS_SQL_fdf0214a24814e13	SYS_SQL_PLAN_24814e134d8f3521	SELECT PRODUCTS.PROD	YES	YES	NO	18
SYS_SQL_fdf0214a24814e13	SYS_SQL_PLAN_24814e1357bbcbf2	SELECT PRODUCTS.PROD	YES	NO	NO	25

Disable Old Plan

```
var ret number
exec :ret := DBMS_SPM.ALTER_SQL_PLAN_BASELINE( -
    sql_handle=>'SYS_SQL_fdf0214a24814e13', -
    plan_name=>'SYS_SQL_PLAN_24814e132c1c9d7b', -
    attribute_name=>’ENABLED', -
    attribute_value=>’NO');
```

EXAMPLE – BETTER PERFORMANCE



BASELINE TIPS

If you can hint it, baseline it (per Tom Kyte)

Alternative to using hints

- **3rd Party Software – can't modify code**
- **Hints difficult to manage over time**
- **Once added, usually forgotten about**

```
SQL> select sql_id, child_number, plan_hash_value, sql_fulltext
     from v$sql
     where sql_text like '%jg%';

SQL_ID        CHILD_NUMBER PLAN_HASH_VALUE SQL_FULLTEXT
------------- ------------ --------------- --------------------------------------------------
12zj3utbrq3kb            0      3021036780 select /* jg */ p.product_name
                                           from order_items o, product p
                                           where o.unit_price

0h9tjus1bgas6            0      3794610757 select /*+ USE_NL(p) +/ /* jg */ p.product_name
                                           from order_items o, product p
                                           wh

SQL> var cnt number
SQL> exec :cnt := dbms_spm.load_plans_from_cursor_cache
    (sql_id => '0h9tjus1bgas6',
     plan_hash_value => 3794610757,
     sql_handle => 'SQL_db5af373d5faa6f5');


SQL> select sql_handle,plan_name,substr(sql_text,1,40) sql_text,
   2 enabled, accepted, fixed, optimizer_cost, to_char(last_executed,'dd-mon-yy HH24:MI') last_executed
   3 from dba_sql_plan_baselines where creator = 'SOE'
   4 order by 1;

SQL_HANDLE                PLAN_NAME                      SQL_TEXT                       ENA ACC FIX
--------------------   ------------------------------   ------------------------------   --- --- ---
SQL_db5af373d5faa6f5   SQL_PLAN_dqqrmfgazp9rp4dcad05d   select /* jg */ p.product_name   NO  YES NO
SQL_db5af373d5faa6f5   SQL_PLAN_dqqrmfgazp9rpc2f36d8b   select /* jg */ p.product_name   YES YES NO
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID  cdgndknbhf0cq, child number 0
-------------------------------------
select /* jg */ p.product_name from order_items o, product p where
o.unit_price = :b1  and o.quantity > :b2  and o.product_id =
p.product_id and p.product_id = :b3

Plan hash value: 3021036780
------------------------------------------------------------------------------
| Id  | Operation                              | Name                |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                     |
|   1 |  MERGE JOIN CARTESIAN                  |                     |
|*  2 |   TABLE ACCESS BY INDEX ROWID          | ORDER_ITEMS         |
|*  3 |    INDEX RANGE SCAN                    | OI_PRODUCT_ID       |
|   4 |   BUFFER SORT                          |                     |
|   5 |    TABLE ACCESS BY INDEX ROWID BATCHED | PRODUCT             |
|*  6 |     INDEX RANGE SCAN                   | PRODUCT_PRODUCT_ID  |
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID  0h9tjus1bgas6, child number 0
-------------------------------------
select /*+ USE_NL(p) +/ /* jg */ p.product_name from order_items o,
product p where o.unit_price = :b1  and o.quantity > :b2  and
o.product_id = p.product_id and p.product_id = :b3

Plan hash value: 3794610757

------------------------------------------------------------------------------
| Id  | Operation                              | Name                |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |                     |
|   1 |  NESTED LOOPS                          |                     |
|   2 |   NESTED LOOPS                         |                     |
|*  3 |    TABLE ACCESS BY INDEX ROWID BATCHED | ORDER_ITEMS         |
|*  4 |     INDEX RANGE SCAN                   | OI_PRODUCT_ID       |
|*  5 |    INDEX RANGE SCAN                    | PRODUCT_PRODUCT_ID  |
|   6 |   TABLE ACCESS BY INDEX ROWID          | PRODUCT             |
```

SQL PATCHES

Can change the execution plans by supplying hints

- **3rd Party Software – can't modify code**

Works in Standard & Enterprise editions

Needs 'grant execute on sys.dbms_sqldiag_internal'

```
-- Command to drop patch if it already exists
exec DBMS_SQLDIAG.DROP_SQL_PATCH('EMP_PATCH');


-- Commands to create patch
begin
  sys.dbms_sqldiag_internal.i_create_patch(
  sql_text => 'select count(*), max(empno) from emp where deptno = :deptno',
  hint_text => 'DYNAMIC_SAMPLING(4)',
  name => 'EMP_PATCH');
end;
/
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID  3zwhs316qhumd, child number 0
-------------------------------------
 select count(*), max(empno) from emp where deptno = :deptno

Plan hash value: 2083865914
------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       |   619 (100)|          |
|   1 |  SORT AGGREGATE    |      |     1 |     7 |            |          |
|*  2 |   TABLE ACCESS FULL| EMP  |   368K|  2521K|   619   (1)| 00:00:01 |
------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------
   2 - filter("DEPTNO"=:DEPTNO)
Note
-----
   - SQL patch "EMP_PATCH" used for this statement
```

```
SQL> select name, category, sql_text, status from dba_sql_patches;

NAME        CATEGORY   SQL_TEXT                       STATUS
----------- ---------- ------------------------------ --------
EMP_PATCH   DEFAULT    select count(*), max(empno)    ENABLED
                       from emp where deptno = :dep
                       tno
```

SQL PATCHES – ANOTHER EXAMPLE

solarwinds

```
begin
  sys.dbms_sqldiag_internal.i_create_patch(
  sql_text => 'select e.empno EOD, e.ename "Employee_name", d.dname "Department",'
              ||' e.hiredate "Date_Hired" from emp e, dept d where d.deptno = :p1'
              ||' and e.deptno = d.deptno',
  hint_text => 'BIND_AWARE',
  name => 'EMP_DEPTNO');
end;
/
```

```
SQL> l
  1  select sql_id,child_number, executions, buffer_gets, sql_text,
  2  is_bind_sensitive, is_bind_aware, is_shareable
  3  from v$sql
  4* where sql_text like 'select e.empno EOD,%'
SQL> /

SQL_ID        CHILD_NUMBER EXECUTIONS BUFFER_GETS SQL_TEXT                      I I I
------------- ------------ ---------- ----------- ------------------------------ - - -
29g1b5n68kyns            0         13      322351 select e.empno EOD, e.ename "E Y N N
29g1b5n68kyns            1         15      140565 select e.empno EOD, e.ename "E Y Y Y
29g1b5n68kyns            2          2       72114 select e.empno EOD, e.ename "E Y Y Y
```

NOTE - SQL patch "EMP_DEPTNO" used for this statement

```
SQL_ID        CHILD_NUMBER EXECUTIONS BUFFER_GETS SQL_TEXT                       I I I
------------- ------------ ---------- ----------- ------------------------------ - - -
29g1b5n68kyns            0         61      571635 select e.empno EOD, e.ename "E Y Y Y
```

SQL PROFILES & TUNING ADVISOR

solarwinds

Tuning Advisor runs nightly – DBMS_AUTO_SQLTUNE

- **Can be configured to automatically implement SQL Profiles**
  - Not recommended
- **Can run manually for advice for one or more SQL statements**

```
DECLARE
  l_sql_tune_task_id  VARCHAR2(100);
BEGIN
  l_sql_tune_task_id := DBMS_SQLTUNE.create_tuning_task ( sql_id => '&sql_id',
    scope => DBMS_SQLTUNE.scope_comprehensive, time_limit  => 60,
    task_name => '&sql_id', description => 'Tuning task for class registration query');
  DBMS_OUTPUT.put_line('l_sql_tune_task_id: ' || l_sql_tune_task_id);
END;
/

EXEC DBMS_SQLTUNE.execute_tuning_task(task_name => '&sql_id');
```

SQL PROFILES & TUNING ADVISOR

solarwinds

Query to report advice

```
VARIABLE auto_rpt CLOB;
 BEGIN
   :auto_rpt :=DBMS_SQLTUNE.REPORT_AUTO_TUNING_TASK ( begin_exec => NULL,  end_exec => NULL, type => 'TEXT'
    , level => 'TYPICAL', section => 'ALL', object_id  => NULL, result_limit => NULL);
   END;
   /
PRINT :auto_rpt
```

```
3- Index Finding (see explain plans section below)
--------------------------------------------------
  The execution plan of this statement can be improved by creating one or more

  indices.

  Recommendation (estimated benefit: 99.78%)
  ------------------------------------------
  - Consider running the Access Advisor to improve the physical schema design
    or creating the recommended index.
    create index SOE.IDX$$_12B40001 on SOE.DEPT("DEPTNO");

  Rationale
  ---------
    Creating the recommended indices significantly improves the execution plan
    of this statement. However, it might be preferable to run "Access Advisor"
    using a representative SQL workload as opposed to a single statement. This

    will allow to get comprehensive index recommendations which takes into
    account index maintenance overhead and additional space consumption.
```

SQL PROFILES – MANUAL SETUP

solarwinds

Select * from table (dbms_xplan.display_cursor('&sql_id,&child,'ADVANCED'));

```
SQL_ID  77qpzs4bt83nc, child number 0
-------------------------------------
select sal salary, e.empno EOD, e.ename "Employee_name", d.dname
"Department", e.hiredate "Date_Hired" from emp e, dept d where d.deptno
= :p1 and e.deptno = d.deptno and sal > :p2 order by sal desc

Plan hash value: 3317977034
-------------------------------------------------------------------------------
| Id  | Operation                              | Name        | Rows  | Bytes |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |             |       |       |
|   1 |  SORT ORDER BY                         |             |     1 |    77 |
|   2 |   MERGE JOIN CARTESIAN                 |             |     1 |    77 |
|*  3 |    TABLE ACCESS FULL                   | DEPT        |     1 |    22 |
|   4 |    BUFFER SORT                         |             |    25 |  1375 |
|*  5 |     TABLE ACCESS BY INDEX ROWID BATCHED| EMP         |    25 |  1375 |
|*  6 |      INDEX RANGE SCAN                  | EMP_DEPTNO  |    25 |       |
-------------------------------------------------------------------------------
Outline Data
-------------
  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('12.1.0.2')
      DB_VERSION('12.1.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$1")
      FULL(@"SEL$1" "D"@"SEL$1")
      INDEX_RS_ASC(@"SEL$1" "E"@"SEL$1" ("EMP"."DEPTNO"))
      BATCH_TABLE_ACCESS_BY_ROWID(@"SEL$1" "E"@"SEL$1")
      LEADING(@"SEL$1" "D"@"SEL$1" "E"@"SEL$1")
      USE_MERGE_CARTESIAN(@"SEL$1" "E"@"SEL$1")
      END_OUTLINE_DATA
  */
etc...
```

```
SQL_ID  1pa2txt6kgbuh, child number 0
-------------------------------------
select /*+ use_hash(e,d) parallel(e, 2) parallel(d, 2) */ sal salary,
e.empno EOD, e.ename "Employee_name", d.dname "Department", e.hiredate
"Date_Hired" from emp e, dept d where d.deptno = :p1 and e.deptno =
d.deptno and sal > :p2 order by sal desc

Plan hash value: 950569827
-------------------------------------------------------------------------------
| Id  | Operation                              | Name        | Rows
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                       |             |
|   1 |  PX COORDINATOR                        |             |
|   2 |   PX SEND QC (ORDER)                   | :TQ10003    |     1
|   3 |    SORT ORDER BY                       |             |     1
|   4 |     PX RECEIVE                         |             |     1
|   5 |      PX SEND RANGE                     | :TQ10002    |     1
|*  6 |       HASH JOIN BUFFERED               |             |     1
|   7 |        PX RECEIVE                      |             |     1
|   8 |         PX SEND HYBRID HASH            | :TQ10000    |     1
|   9 |          STATISTICS COLLECTOR          |             |
|  10 |           PX BLOCK ITERATOR            |             |     1
|* 11 |            TABLE ACCESS FULL           | DEPT        |     1
|  12 |         PX RECEIVE                     |             |    25
|  13 |          PX SEND HYBRID HASH           | :TQ10001    |    25
|  14 |           PX SELECTOR                  |             |
|* 15 |            TABLE ACCESS BY INDEX ROWID BATCHED| EMP  |    25
|* 16 |             INDEX RANGE SCAN           | EMP_DEPTNO  |    25
-------------------------------------------------------------------------------
Outline Data
-------------
  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('12.1.0.2')
      DB_VERSION('12.1.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$1")
      FULL(@"SEL$1" "D"@"SEL$1")
      INDEX_RS_ASC(@"SEL$1" "E"@"SEL$1" ("EMP"."DEPTNO"))
      BATCH_TABLE_ACCESS_BY_ROWID(@"SEL$1" "E"@"SEL$1")
      LEADING(@"SEL$1" "D"@"SEL$1" "E"@"SEL$1")
      USE_HASH(@"SEL$1" "E"@"SEL$1")
      PQ_DISTRIBUTE(@"SEL$1" "E"@"SEL$1" HASH HASH)
      END_OUTLINE_DATA
  */
etc...
```

SQL PROFILES – MANUAL SETUP

```
DECLARE
    sqlt            clob;
    BEGIN
    sqlt := q'!select sal salary,
                e.empno EOD, e.ename "Employee_name", d.dname "Department",
                e.hiredate "Date_Hired"
                from emp e, dept d
                where d.deptno = :p1 and e.deptno = d.deptnoand sal > :p2
                order by sal desc!';
    dbms_sqltune.import_sql_profile( sql_text => sqlt,
        name => 'SP_EMPHASH',
        profile => sqlprof_attr(q'!PQ_DISTRIBUTE(@"SEL$1" "E"@"SEL$1" BROADCAST NONE)!',
        q'!USE_HASH(@"SEL$1" "E"@"SEL$1")!',
        q'!LEADING(@"SEL$1" "D"@"SEL$1" "E"@"SEL$1")!',
        q'!FULL(@"SEL$1" "E"@"SEL$1")!',
        q'!FULL(@"SEL$1" "D"@"SEL$1")!',
        q'!OUTLINE_LEAF(@"SEL$1")!',
        q'!ALL_ROWS!',
        q'!DB_VERSION('12.1.0.2')!',
        q'!OPTIMIZER_FEATURES_ENABLE('12.1.0.2')!',
        q'!IGNORE_OPTIM_EMBEDDED_HINTS!'),
         force_match => true );
    END;
    /
```

```
SQL_ID  77qpzs4bt83nc, child number 1
-------------------------------------
select sal salary, e.empno EOD, e.ename "Employee_name", d.dname
"Department", e.hiredate "Date_Hired" from emp e, dept d where d.deptno
= :p1 and e.deptno = d.deptno and sal > :p2 order by sal desc

Plan hash value: 3357797783

-------------------------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |       |       |  3358 (100)|          |
|   1 |  SORT ORDER BY      |      |    26 |   988 |  3358   (1)| 00:00:01 |

|*  2 |   HASH JOIN         |      |    26 |   988 |  3357   (1)| 00:00:01 |
|*  3 |    TABLE ACCESS FULL| DEPT |     1 |    13 |  2739   (1)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| EMP  |    26 |   650 |   619   (1)| 00:00:01 |
-------------------------------------------------------------------------------

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

   2 - access("E"."DEPTNO"="D"."DEPTNO")
   3 - filter("D"."DEPTNO"=:P1)
   4 - filter(("E"."DEPTNO"=:P1 AND "SAL">:P2))

Note

-----
   - SQL profile SQLPROFILE_EMPHASH used for this statement
```

SUMMARY

solarwinds

Oracle has many ways to improve Plan Stability

- **Almost too many**
  - It can be confusing at times
  - Outlines, Profiles, Patches & Baselines

Plans change for many reasons

- **Data grows, system changes, stale statisics**
- **Oracle features – such as SQL Directives, Adaptive Cursor Sharing, etc…**
- **V$SQL_PLAN & V$SQL_SHARED_CURSOR helps with finding out the ‘WHY’**
  - Don’t forget about XML Query of column ‘Other_XML’

Monitor the features that may impact plans

- **dba_sql_plan_directives, dba_sql_plan_dir_objects, & v$sql_reotimization_hints**

Consider using Baselines, Patches & Profile to help stabilize plans

Q & A

RESOLVE PERFORMANCE ISSUES QUICKLY

solarwinds

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
- Agent-less architecture with no dependence on Oracle® Packs; installs in minutes

www.solarwinds.com/dpa-download/