



Ignite IT Performance™

Hey Oracle Optimizer! Don't Mess with MY Plans

Janis Griffin
Senior DBA, Confio Software

Who Am I?

- Senior DBA for Confio Software
 - JanisGriffin@confio.com
 - Twitter - @[DoBoutAnything](https://twitter.com/DoBoutAnything)
 - Current – 24+ Years in Oracle
 - DBA and Developer
- Specialize in Performance Tuning
- Review Database Performance for Customers and Prospects
- Confio Software
 - Makers of Ignite8 Response Time Analysis Tools
 - IgniteVM for Oracle/SQL/Sybase/DB2 on Vmware
 - AlarmVm for VM Administrators

- History of Plan Stability
 - Outlines
 - Profiles
- New in 11.1 – SQL Plan Management (SPM)
 - How does it work – baselines
 - SPM main components & system views
 - dbms_spm
- Several Examples
 - Customer Query
 - Product Query
- Q & A

- Oracle 8 – Introduced cost-based optimizer
 - Allowed for:
 - Hash joins & histograms
 - Partitioned tables & 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 11.1 – SQL Plan Management
 - Free – No Extra \$\$\$ with Enterprise
 - alter system set control_management_pack_access = 'NONE'; -- disables DIAG/Tuning
 - DBMS_SPM & Baselines

■ 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 & index is dropped)
 - `DBMS_OUTLN` / `alter session set create_stored_outlines = true;`

■ SQL Profiles

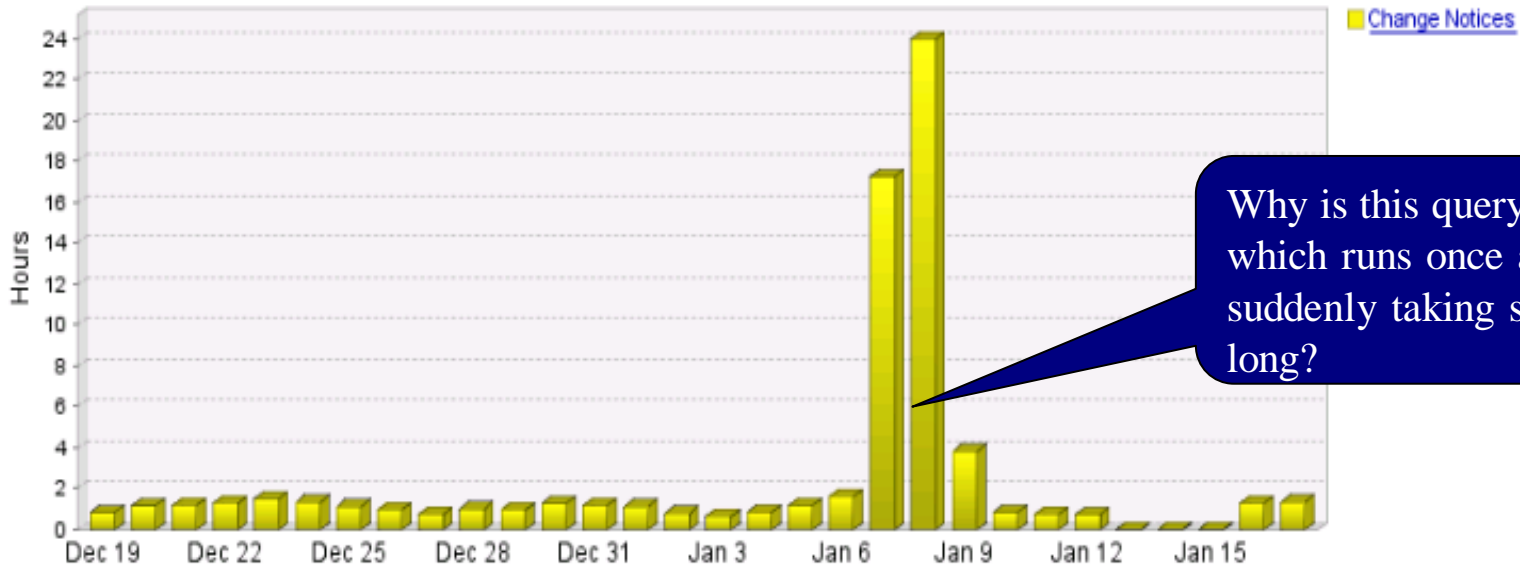
- Created by SQL Tuning Advisor (`dbms_sqltune` - cost \$\$\$)
- Similar to Outlines – implemented with hints
- Uses `OPT_ESTIMATE` hint – not always accurate
 - Tries to improve cost estimates over time (factors 10x estimate)
- Nightly look at SQLs to find better execution plan

- 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 11g (still work)
- Profiles/Tuning Advisor – Cost \$\$\$

- How Oracle manages plan stability in 11g
 - Tries to prevent performance regressions resulting from sudden changes in execution plans
- Examples of unpredictable changes in plans:
 - New optimizer version
 - Changes in optimizer statistics and/or parameters
 - Changes to schema and metadata definitions
 - E.g. Dropping an index, Data growing, Statistics stale
 - Changes to system settings
- Common Uses of SPM
 - System & Data changes causing performance regressions
 - Database Upgrades & New Application Installs

Why SQL Plan Management (SPM)

Specified SQL Statements | PROD_STA740
December 19, 2009 to January 17, 2010



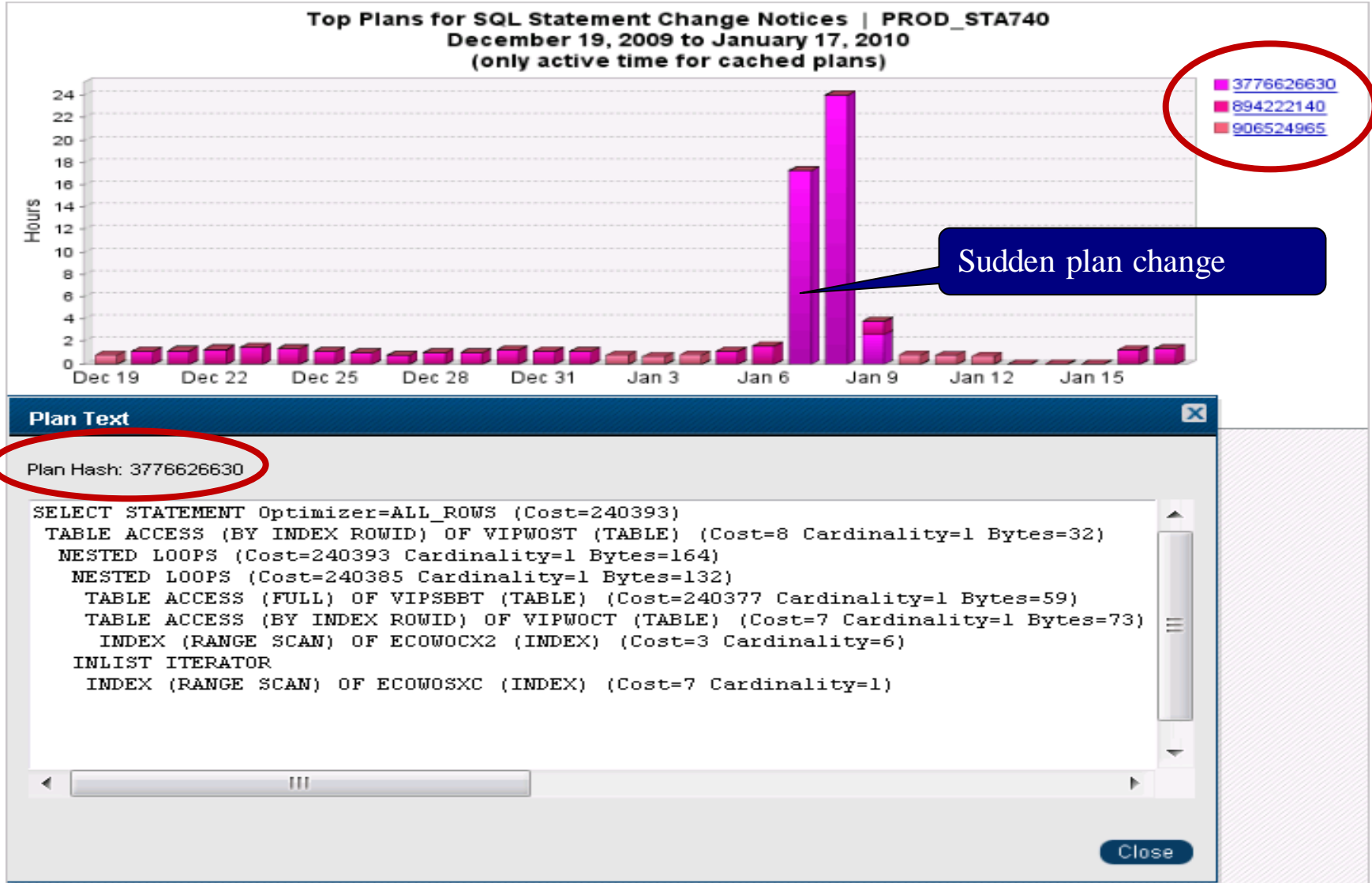
Why is this query, which runs once a day, suddenly taking so long?

[Hide Full SQL Text](#)

Full SQL Text

SQL Hash Name	SQL Text
Change Notices	<pre> 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" </pre>

Why SQL Plan Management (SPM)



- Preventative Mechanism for Plan Stability
 - Optimizer records & evaluates execution plans over time
 - **SQL plan baselines** - a set of existing plans that are efficient
 - Baselines can evolve overtime for better performance
 - Preserves performance regardless of changes
 - DBA can verify that only comparable or better plans will be used

- To use SPM - two init.ora parameters
 - **optimizer_capture_sql_plan_baselines**
 - Controls auto-capture of SQL plan baselines for repeatable statements
 - Set to false by default in 11gR1
 - **optimizer_use_sql_plan_baselines**
 - Controls the use of existing SQL plan baselines by the optimizer
 - Set to true by default in 11gR1

- Manage SQL Plan Baselines with EM or dbms_spm pkg
 - Can use Tuning Advisor to automatically test / verify changes (however, extra \$\$\$)

■ Baseline Capture

- Records historical plans for SQL statements
- Does not record ad hoc sqls – sql has to run at least twice
- Auto-capture
 - `alter <session/system> set optimizer_capture_sql_plan_baselines=TRUE;`

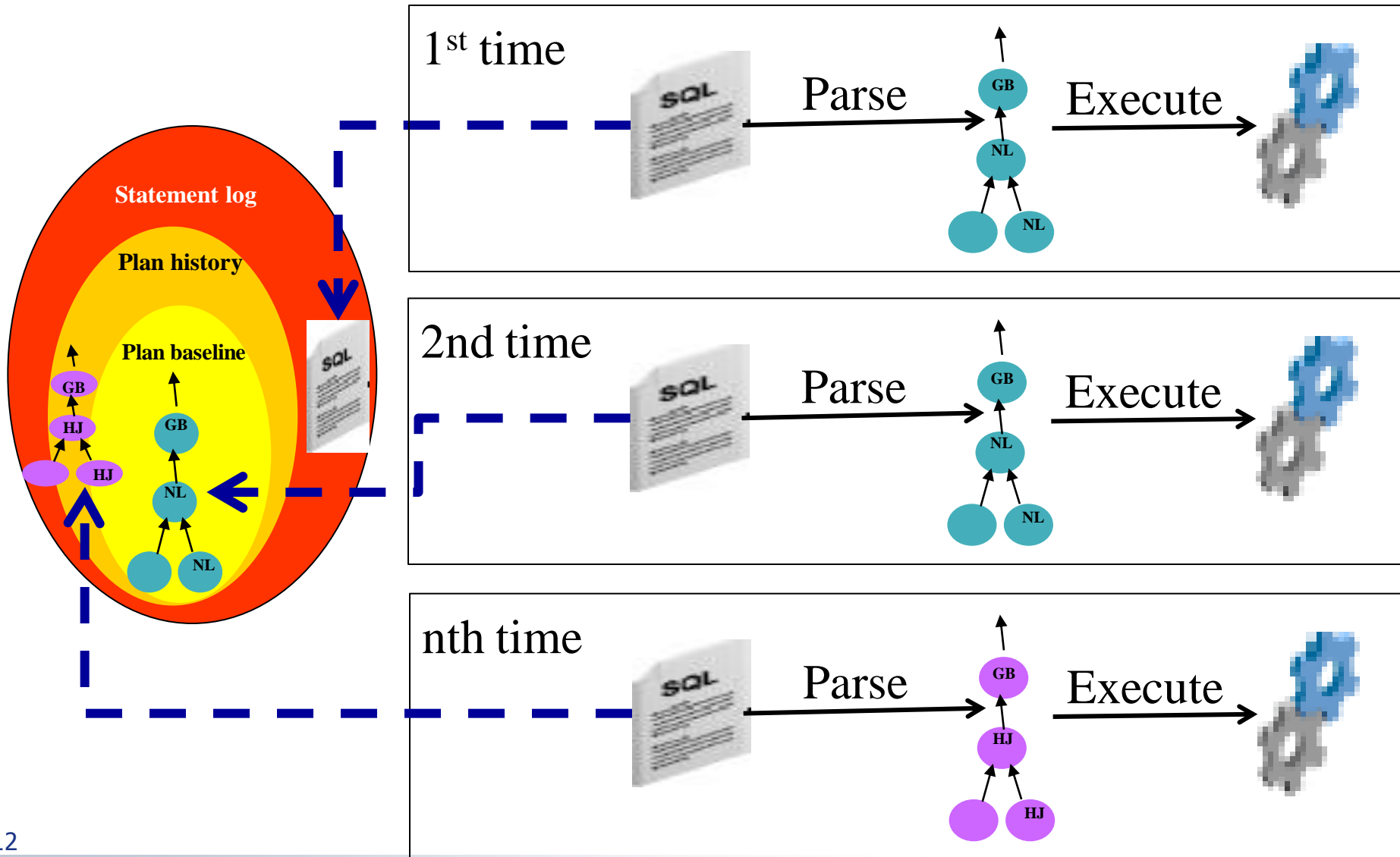
■ Baseline Selection

- Optimizer looks at stored plan history
- Tries to prevent potential performance regressions
- Unaccepted versus Accepted

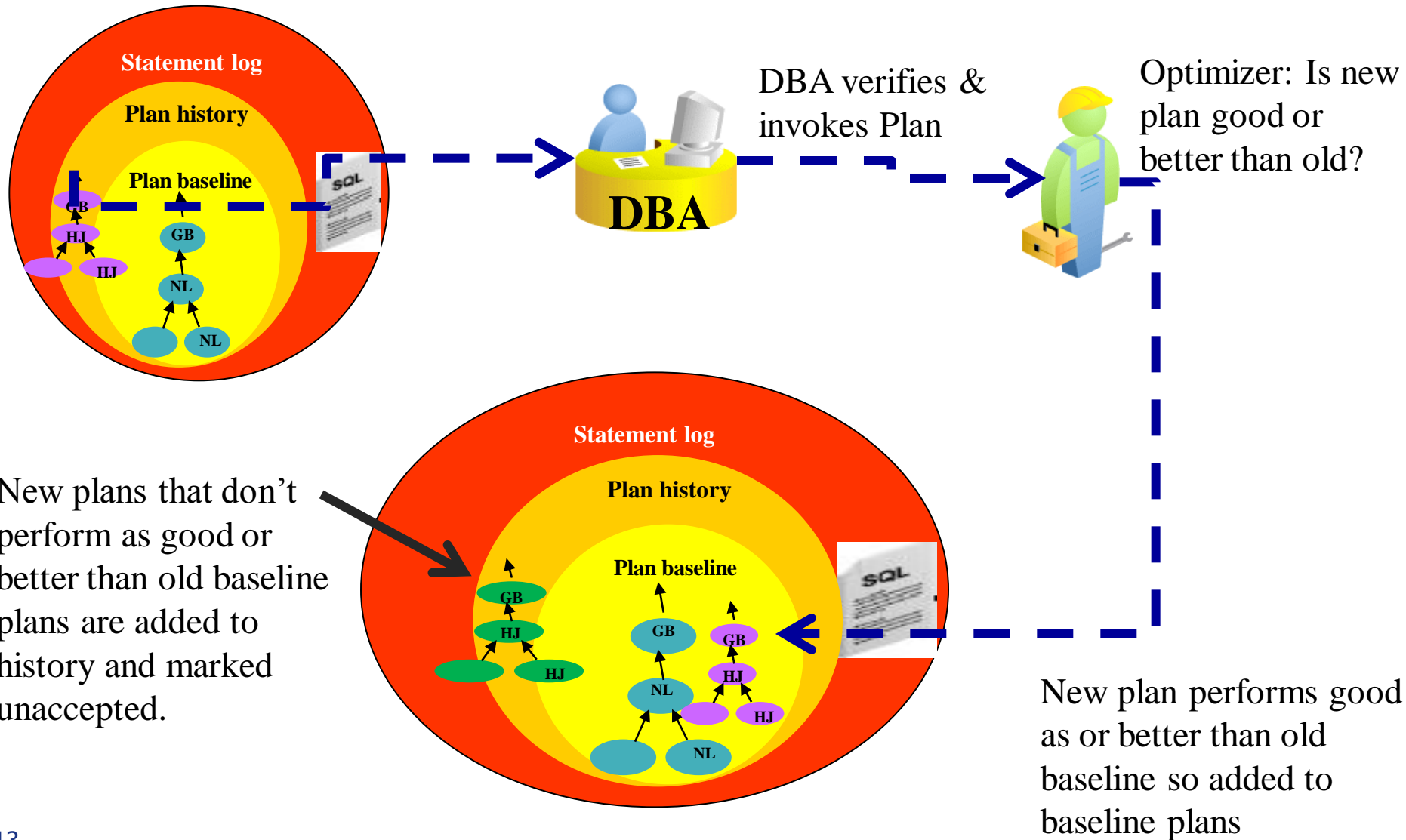
■ Baseline Evolution

- Optimizer evaluates the performance of new plans
- Integrate plans with better performance into SQL plan baselines.

How SPM Works - Visually



How SPM Works – cont.



- SQL Management Base (SMB) – SYSAUX tablespace
- Stores Statement Log, Plan Histories & Plan Baselines
 - Also, contains SQL Profiles & Outlines
- Purge runs weekly
 - Disk Space Quota: default -10% of SYSAUX,
 - ranges: 1-50%
 - Plan Retention: default - 53 weeks since last used
 - ranges: 5 wks – 523 wks (10 yrs+)
- Tables & Views
 - DBA_SQL_MANAGEMENT_CONFIG
 - SMB\$CONFIG
 - DBA_SQL_PLAN_BASELINES
- Packages - DBMS_SPM & DBMS_SPM_INTERNAL

- Manage one or more plans
 - LOAD_PLANS_FROM_CURSOR_CACHE
 - ALTER_SQL_PLAN_BASELINE ('fix' or 'enable')
 - EVOLVE_SQL_PLAN_BASELINE ('verify' and/or 'commit')
 - DROP_SQL_PLAN_BASELINE

- Load Baselines from Tuning Sets / Other Sources
 - LOAD_PLANS_FROM_SQLSET
 - CREATE_STGTAB_BASELINE
 - PACK_STGTAB_BASELINE (export)
 - UNPACK_STGTAB_BASELINE (import)

- SMB Configuration
 - CONFIGURE - Disk Space & Retention in SMB\$CONFIG

Example 1 – Customer Query

```
ALTER session SET optimizer_capture_sql_plan_baselines=TRUE;
```

```
DECLARE
```

```
  cnt NUMBER;
```

```
BEGIN
```

```
  FOR i IN 1..1000000 LOOP
```

```
    SELECT count(*) INTO cnt
```

```
    FROM orders a, customers b
```

```
    WHERE a.customer_id = b.customer_id;
```

```
  END LOOP;
```

```
END;
```

```
show parameter baselines
```

NAME	TYPE	VALUE
optimizer_capture_sql_plan_baselines	boolean	TRUE
optimizer_use_sql_plan_baselines	boolean	TRUE

Alternative to 'ALTER session':

```
SELECT sql_id, plan_hash_value from v$sql where sql_text like 'SELECT count(*)%';
```

```
VAR ret_var NUMBER
```

```
EXEC :ret_var := dbms_spm.load_plans_from_cursor_cache( SQL_ID=>'&sql_id',  
  PLAN_HASH_VALUE=>&plan_hash_value, FIXED=>'&fixed', ENABLED=>'&enabled');
```

Example 1 – Review Baselines

Baselines created from logon trigger for SOE user:

```
SELECT sql_handle,  
       plan_name,  
       SUBSTR(sql_text,1,40) sql_text,  
       enabled, accepted, fixed, optimizer_cost,  
       to_char(last_executed,'dd-mon-yy HH24:MI') last_executed  
FROM dba_sql_plan_baselines  
WHERE creator = 'SOE'  
ORDER BY 1
```

SQL_HANDLE	PLAN_NAME	SQL_TEXT	ENA	ACC	FIX	OPTIMIZER_COST
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d78e1961cee	select count(*) from orders a, customers	YES	YES	NO	19309
SYS_SQL_9c3c4291df2a9446	SYS_SQL_PLAN_df2a9446ed88afee	SELECT ATTRIBUTE,SCOPE,NUMERIC_VALUE,CHA	YES	YES	NO	2
SYS_SQL_e744325067d2db2f	SYS_SQL_PLAN_67d2db2fed88afee	SELECT CHAR_VALUE FROM SYSTEM.PRODUCT_PR	YES	YES	NO	2

Example 1 – Show Plan

DBMS_XPLAN – New Baseline Function:

```
select * from
```

```
table(dbms_xplan.display_sql_plan_baseline(sql_handle=>'&SQL_HANDLE'));
```

```
PLAN_TABLE_OUTPUT
-----
SQL handle: SYS_SQL_547c574c74755d78
SQL text: select count(*) from orders a, customers b where a.customer_id =
         b.customer_id
-----

Plan name: SYS_SQL_PLAN_74755d78e1961cee
Enabled: YES   Fixed: NO   Accepted: YES   Origin: AUTO-CAPTURE
-----

Plan hash value: 1706270605

-----
| Id | Operation                               | Name                | Rows  | Bytes | TempSpcl | Cost (%CPU) | Time |
-----|-----|-----|-----|-----|-----|-----|-----|-----|
|  0 | SELECT STATEMENT                        |                     |      1 |      12 |           | 19309 (1)   | 00:03:52 |
|  1 |   SORT AGGREGATE                        |                     |      1 |      12 |           |           |         |
| * 2 |    HASH JOIN                             |                     | 5707K | 65M   | 81M      | 19309 (1)   | 00:03:52 |
|  3 |      INDEX FAST FULL SCAN                | CUSTOMERS_PK        | 4750K | 27M   |           | 4653 (1)   | 00:00:56 |
|  4 |      INDEX FAST FULL SCAN                | ORD_CUSTOMER_IX     | 5707K | 32M   |           | 5700 (1)   | 00:01:09 |
-----

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

   2 - access("A"."CUSTOMER_ID"="B"."CUSTOMER_ID")
```

Example 1 – Manage Baselines

```
DROP INDEX ORD_CUSTOMER_IX; -- run query several more times
```

```
SELECT sql_handle, plan_name, ..., enabled, accepted, fixed, etc...
```

```
FROM dba_sql_plan_baselines
```

```
where sql_handle = 'SYS_SQL_547c574c74755d78';
```

SQL_HANDLE	PLAN_NAME	SQL_TEXT	ENA	ACC	FIX	OPTIMIZER_COST
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d78e1961cee	select count(*) from orders a, customers	YES	YES	NO	19309
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d785409a514	select count(*) from orders a, customers	YES	NO	NO	25440

```
VAR ret_var CLOB
```

```
EXEC :ret_var := dbms_spm.evolve_sql_plan_baseline( -  
SQL_HANDLE=> '&sql_handle', PLAN_NAME=> '&plan_name', -  
VERIFY=> '&verify', -  
COMMIT=> '&commit');
```

SQL_HANDLE	PLAN_NAME	SQL_TEXT	ENA	ACC	FIX	OPTIMIZER_COST
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d78e1961cee	select count(*) from orders a, customers	YES	YES	NO	19309
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d785409a514	select count(*) from orders a, customers	YES	YES	NO	25440

Example 1 – Manage Baselines

```
select * from table(dbms_xplan.display_sql_plan_baseline( -  
  sql_handle=>'SYS_SQL_547c574c74755d78', -  
  plan_name=>'SYS_SQL_PLAN_74755d785409a514'))
```

PLAN_TABLE_OUTPUT

```
-----  
SQL handle: SYS_SQL_547c574c74755d78  
SQL text: select count(*) from orders a, customers b where a.customer_id =  
         b.customer_id  
-----
```

```
Plan name: SYS_SQL_PLAN_74755d785409a514  
Enabled: YES      Fixed: NO      Accepted: YES      Origin: AUTO-CAPTURE  
-----
```

```
Plan hash value: 2049750053  
-----
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (<%CPU>)	Time
0	SELECT STATEMENT		1	12		25499 (1)	00:05:06
1	SORT AGGREGATE		1	12			
* 2	HASH JOIN		5734K	65M	81M	25499 (1)	00:05:06
3	INDEX FAST FULL SCAN	CUSTOMERS_PK	4750K	27M		4653 (1)	00:00:56
4	TABLE ACCESS FULL	ORDERS	5734K	32M		11866 (1)	00:02:23

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

```
2 - access("A"."CUSTOMER_ID"="B"."CUSTOMER_ID")
```

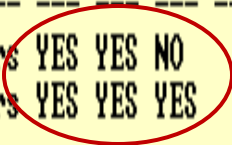
Example 1 – Fixed Baselines

'FIX' the baseline so that it does NOT evolve over time
(A fixed plan takes precedence over a non-fixed plan)

```
VAR ret_var NUMBER  
EXEC :ret_var := dbms_spm.alter_sql_plan_baseline( -  
SQL_HANDLE=>'&sql_handle', PLAN_NAME=>'&plan_name', -  
ATTRIBUTE_NAME=>'&fixed_or_enabled', -  
ATTRIBUTE_VALUE=>'&yes_or_no');
```

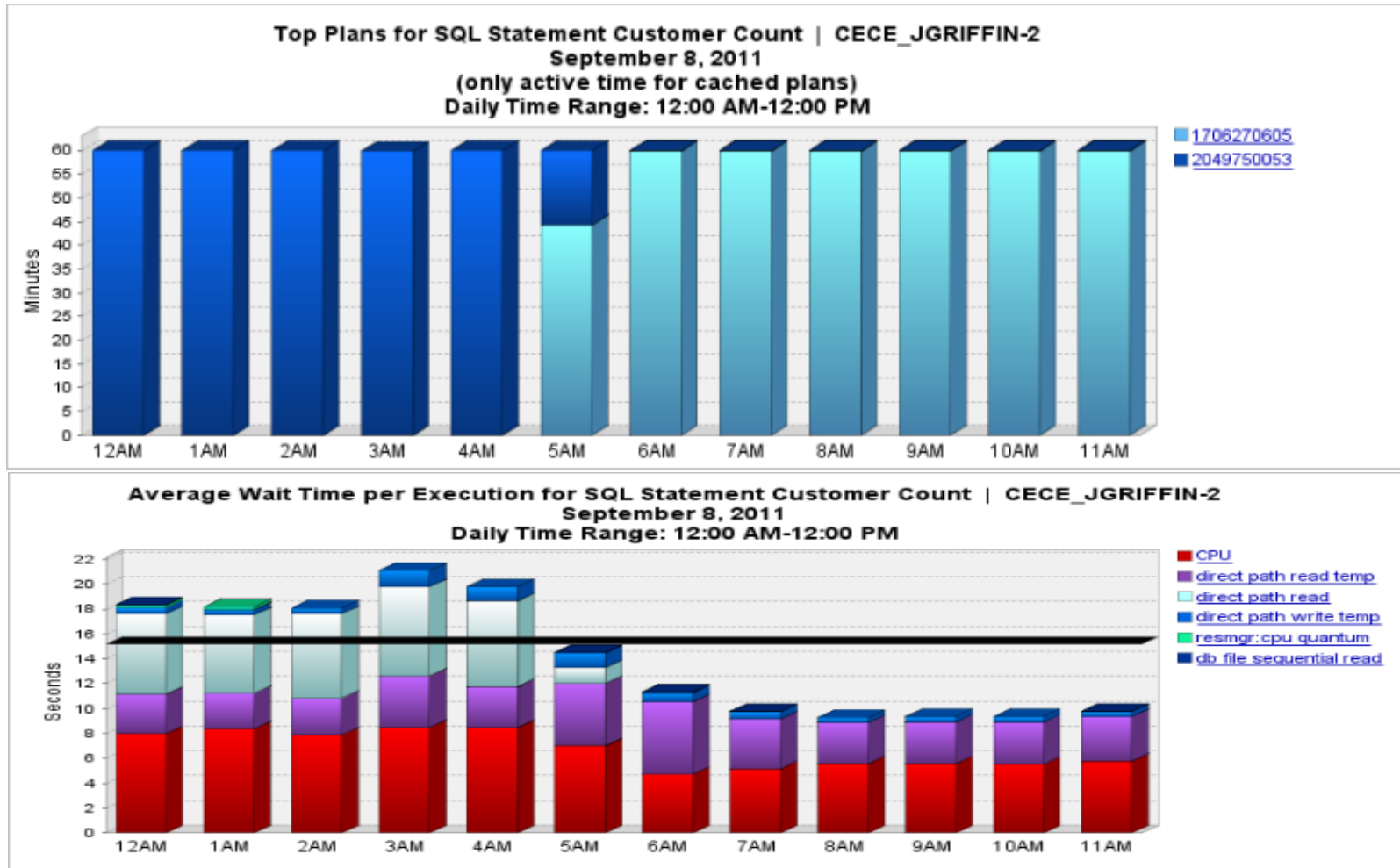
'5409a514'
'FIXED'
'YES'

SQL_HANDLE	PLAN_NAME	SQL_TEXT	ENA	ACC	FIX	OPTIMIZER_COST
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d78e1961cee	select count(*) from orders a, customers	YES	YES	NO	19309
SYS_SQL_547c574c74755d78	SYS_SQL_PLAN_74755d785409a514	select count(*) from orders a, customers	YES	YES	YES	25440



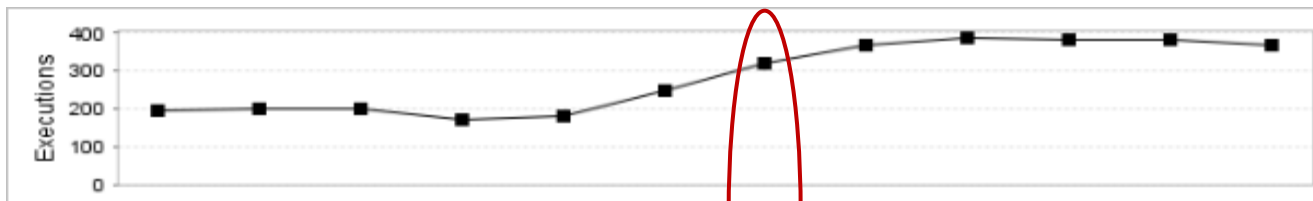
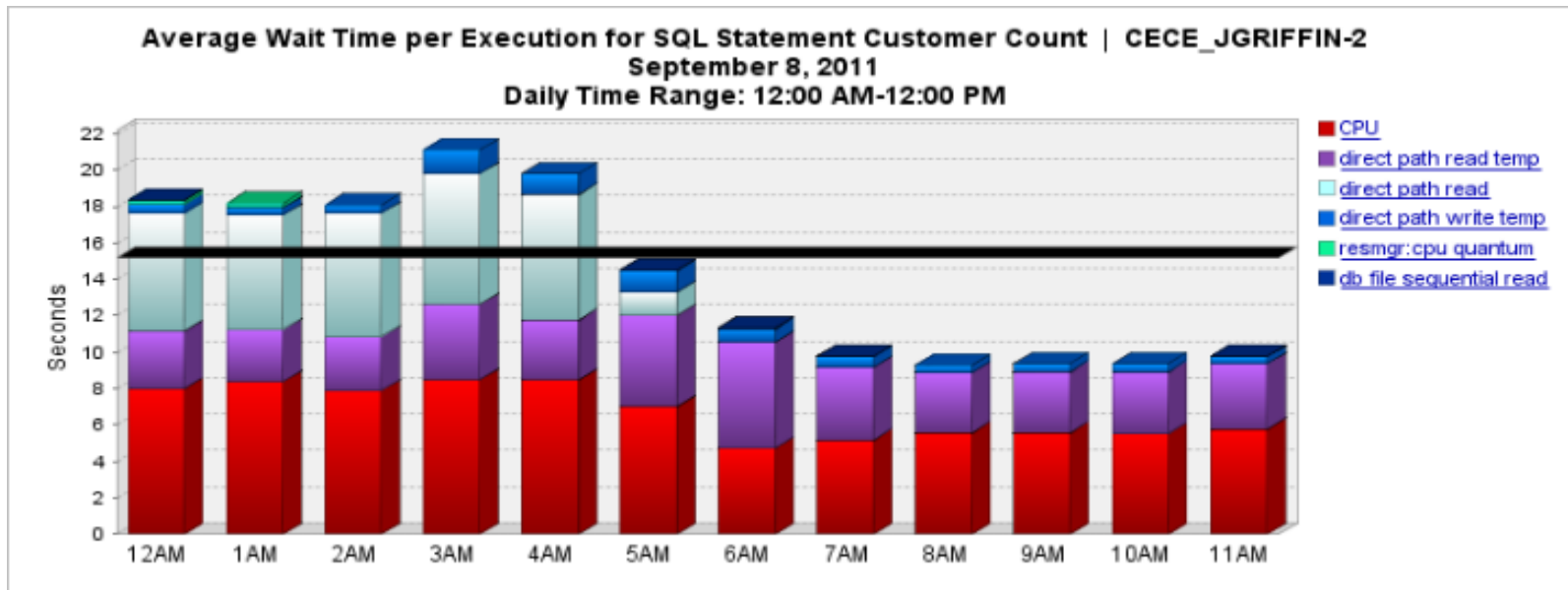
Example 1 – Performance Plans

Fixed Bad Baseline:

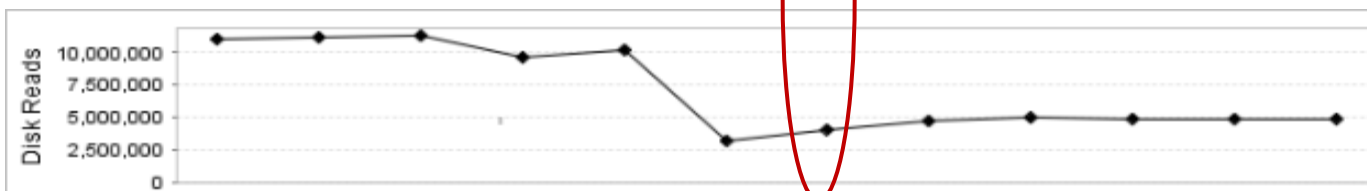


Example 1 - Performance

Disabled Bad Baseline:



Almost double the executions in half the time.



Approximately, 1/3 less disk reads.

Reporting on Evolving Plans

```
SET SERVEROUTPUT ON LONG 10000
```

```
DECLARE rpt clob;
```

```
BEGIN
```

```
rpt := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE( -
```

```
  sql_handle => '&sql_handle', -
```

```
  plan_name => '&plan_name', -
```

```
  verify => '&verify', -
```

```
  commit => '&commit');
```

```
DBMS_OUTPUT.PUT_LINE(rpt);
```

```
END;
```

Evolve SQL Plan Baseline Report			
Inputs:			
SQL_HANDLE	=	SYS_SQL_97036e1ea811e28c	
PLAN_NAME	=	SYS_SQL_PLAN_a811e28c5c25116f	
TIME_LIMIT	=	DBMS_SPM.AUTO_LIMIT	
VERIFY	=	yes	
COMMIT	=	no	
Plan: SYS_SQL_PLAN_a811e28c5c25116f			
Plan was			
verified: Time used .187 seconds.			
Failed performance criterion: Compound improvement ratio < .67			
	<u>Baseline Plan</u>	<u>Test Plan</u>	<u>Improv. Ratio</u>
Execution Status:	COMPLETE	COMPLETE	
Rows Processed:	14	14	
Elapsed Time(ms):	18	120	.15
CPU Time(ms):	31	46	.67
Buffer Gets:	2509	11064	.23
Disk Reads:	0	37	0
Direct Writes:	0	0	
Fetches:	0	12	0
Executions:	1	1	
Report			
Summary			
Number of SQL plan baselines verified: 1.			
Number of SQL plan baselines evolved: 0.			

Bug – Verifying Plans

```
Evolve SQL Plan Baseline Report
-----
Inputs:
-----
SQL_HANDLE =
PLAN_NAME  =
TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
VERIFY     = YES
COMMIT     = NO

Plan: SYS_SQL_PLAN_24814e13070fd4bf
-----
Plan was verified: Time used .058 seconds.
Plan verification
encountered an error (ORA-1008).
ORA-01008: not all variables boun

          Baseline Plan          Test Plan          Improv. Ratio
-----
Execution Status:          PARTIAL          PARTIAL
Rows Processed:           0              0
Elapsed Time(ms):         0              0
CPU Time(ms):             0              0
Buffer Gets:              0              0
Disk Reads:               0              0
Direct Writes:            0              0
Fetches:                  0              0
Executions:               0              0
```

- Bug 9913823: ORA-1008 WITH DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE
- Queries using Bind Variables - Fixed in 11.2.0.2.0
- Workaround – set undocumented parameter to value between 400 (default) & 3999
alter system set "_cursor_bind_capture_area_size" = 3350;
- Need to remove captured baselines so Optimizer will capture the bind variables again

Removing Baselines

```
declare
  stmt varchar2(4000);
cursor get_base is select sql_handle, plan_name from dba_sql_plan_baselines;
begin
for get_rec in get_base loop
stmt := DBMS_SPM.DROP_SQL_PLAN_BASELINE
      (get_rec.sql_handle,get_rec.plan_name);
end loop;
end;
```

```
declare
  stmt varchar2(4000);
begin
stmt := DBMS_SPM.DROP_SQL_PLAN_BASELINE ('&sql_handle','&plan_name');
end;
```

Example 2 – 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
```

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_24814e1357bbc2	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', verify => 'YES', commit => 'NO');  
  DBMS_OUTPUT.PUT_LINE(rpt);  
END;
```

Example 2 – 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 2 – High Cost

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 2 – Better Plan?

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_24814e134d8f3521'))
```

```
Plan name: SYS_SQL_PLAN_24814e134d8f3521
Enabled: YES      Fixed: NO      Accepted: NO      Origin: AUTO-CAPTURE
```

PLAN_TABLE_OUTPUT

```
Plan hash value: 1569745754
```

Id	Operation	Name	Rows	Bytes	Cost
0	SELECT STATEMENT		107	340K	18
* 1	COUNT STOPKEY				
* 2	HASH JOIN OUTER		107	340K	18
* 3	HASH JOIN		107	124K	5
4	TABLE ACCESS BY INDEX ROWID	PRODUCT_INFORMATION	49	56497	2
* 5	INDEX RANGE SCAN	PROD_CATEGORY_IX	20		1
6	TABLE ACCESS BY INDEX ROWID	INVENTORIES	8982	342K	2
* 7	INDEX RANGE SCAN	INVENTORIES_IX1	3593		1
* 8	TABLE ACCESS FULL	PRODUCT_DESCRIPTIONS	49	99K	11

Predicate Information (identified by operation id):

```

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

Example 2 – Force Plan

- Force Evolution of new baseline

```
SET SERVEROUTPUT ONLONG 10000
```

```
DECLARE rpt clob;
```

```
BEGIN
```

```
rpt := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE( -
```

```
  sql_handle=>'SYS_SQL_fdf0214a24814e13', plan_name=>'SYS_SQL_PLAN_24814e134d8f3521', -  
  verify=>'NO', commit=>'YES');
```

```
DBMS_OUTPUT.PUT_LINE(rpt);
```

```
END;
```

- Disable Old Plan

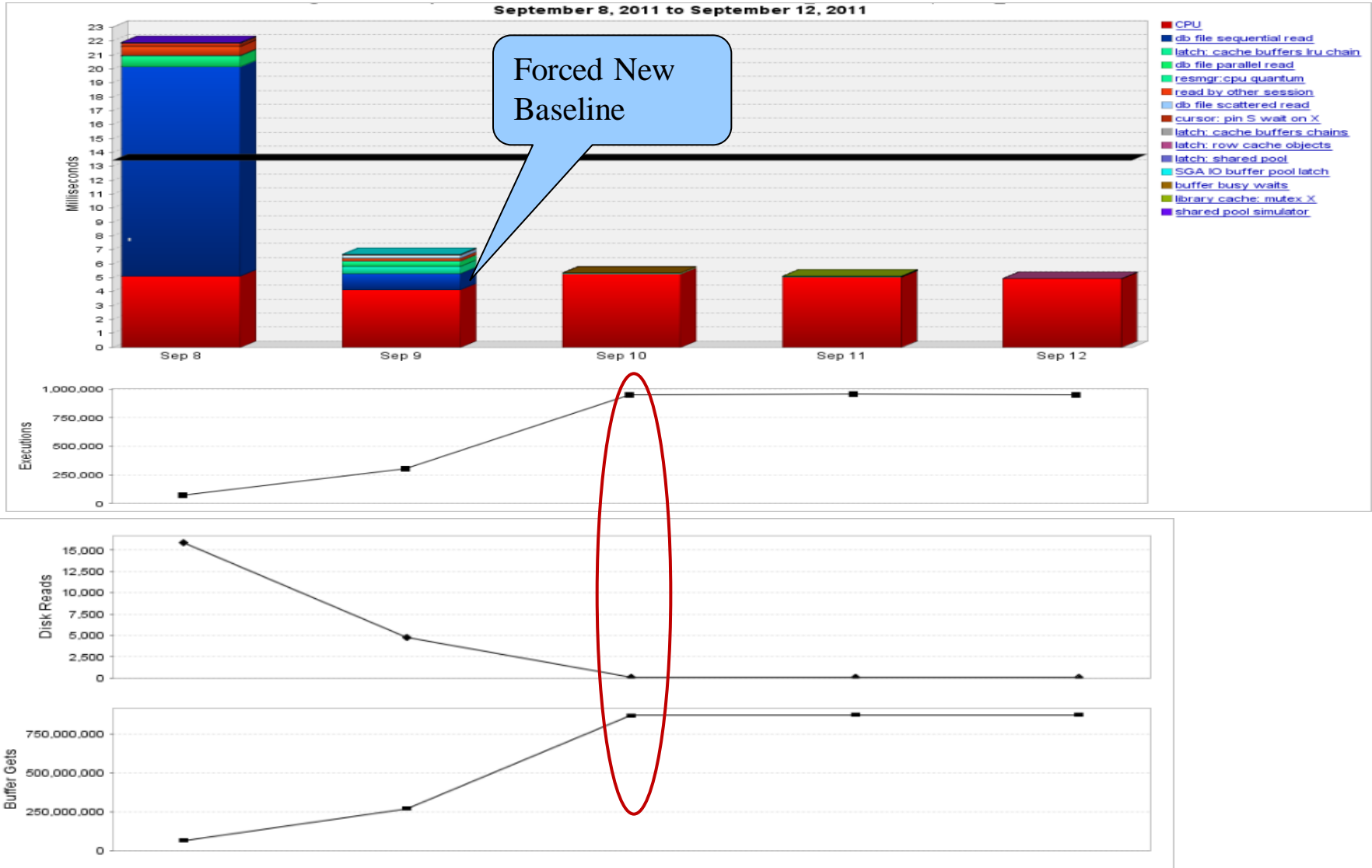
```
var ret number
```

```
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');
```

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_24814e1357bbc2	SELECT PRODUCTS.PROD	YES	NO	NO	25

Example 2 – Better Performance



- SPM improves Plan Stability using baselines
 - Reduces performance regression
 - By choosing only better plans when things change
- Allows the optimizer to capture, select and evolve the better plans overtime
 - Automatically, via Tuning Packs
 - Or, DBA controlled, via DBMS_SPM
- Optimizer takes a very conservative approach when evolving plan.
 - Still requires DBA intervention

- Award Winning Performance Tools
 - Ignite8 for SQL Server, Oracle, DB2, Sybase
 - IgniteVM for Databases on Vmware
- Download at www.confio.com
- Provides Answers for
 - What changes recently that affected end users?
 - Which plan performs better over time?
 - Who and how should we fix the problem?

Download free trial at

www.confio.com