SQL Plan Management to the Rescue!

GLOC 2014

Craig Martin

Who am I?

- Applications DBA for Nationwide in Columbus, OH
- Worked with SQL and PL/SQL on Oracle since 2002

Twitter: @c_martin2

Email: cmartin2@cmartin2.com

Website: http://cmartin2.com

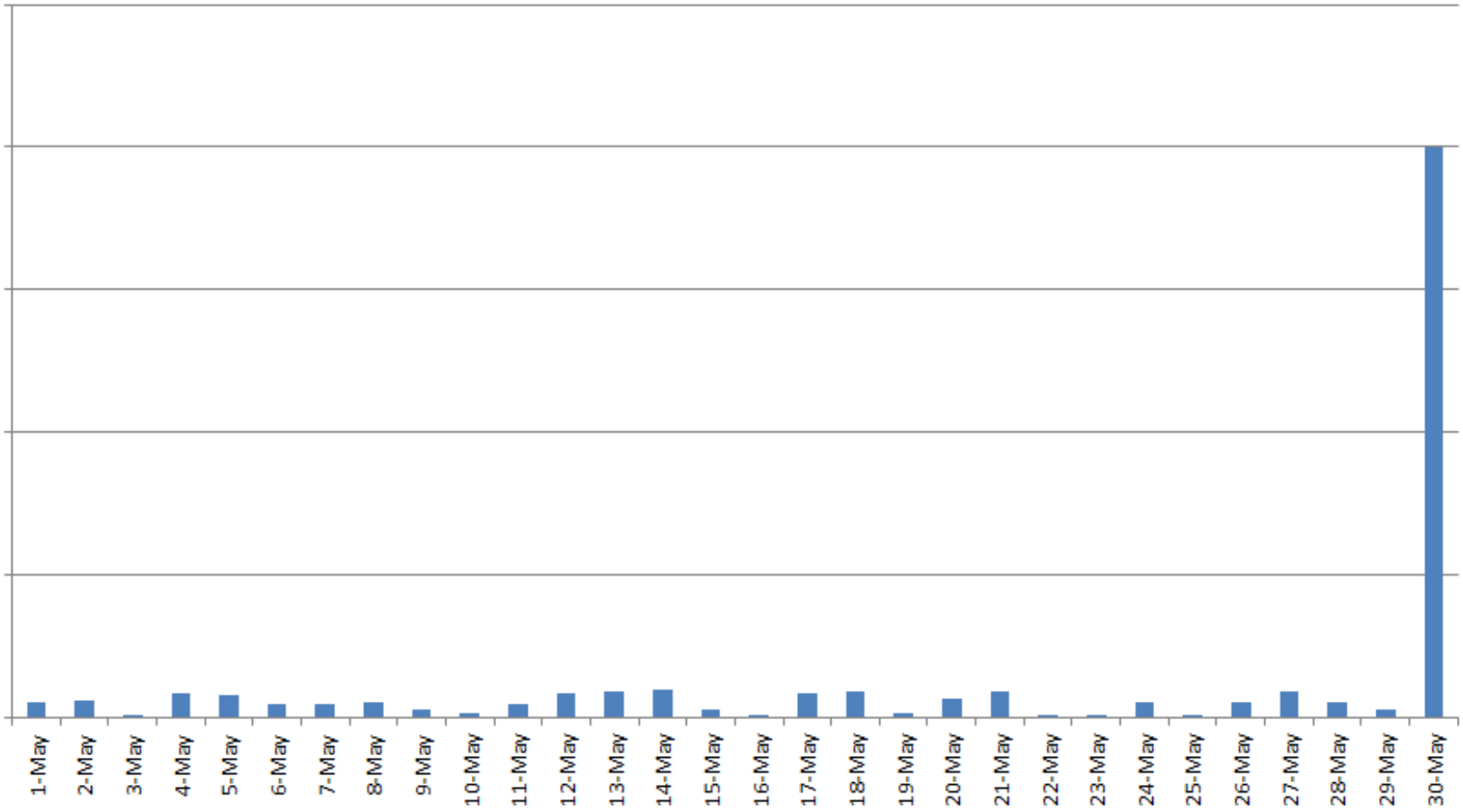
Nationwide Disclaimer

- The opinions I share are mine, and do not necessarily reflect those of Nationwide
- Nationwide does not endorse any process or product mentioned
- It is up to each individual to evaluate the content in this presentation and do their own testing
- Use this information at your own risk


© London Media

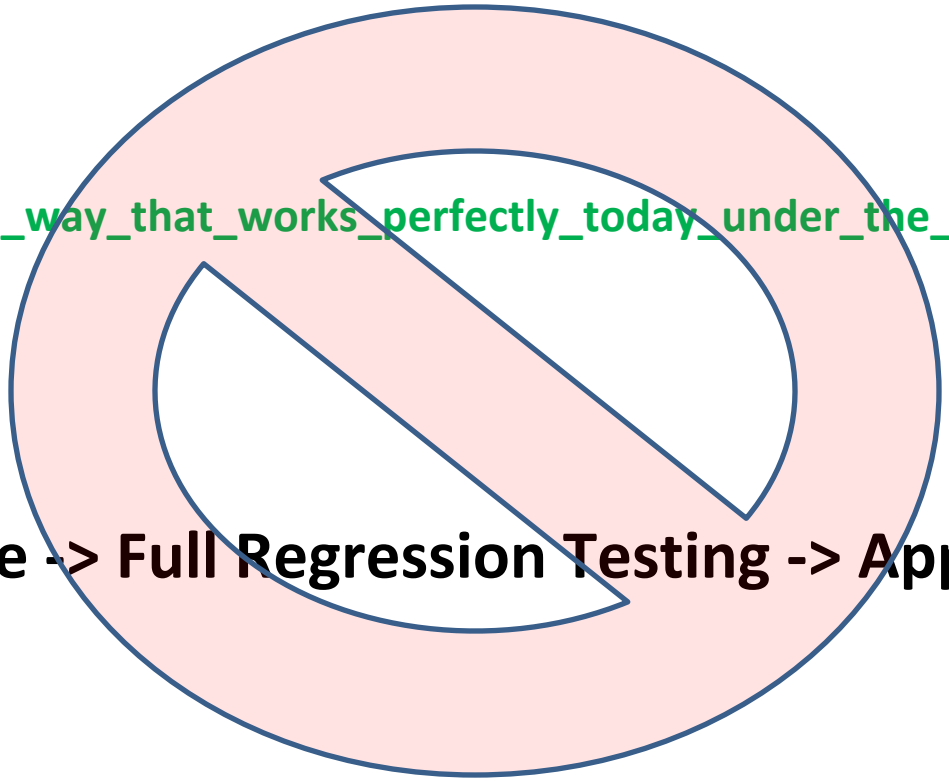

GO BACK
TO BED!


SPM

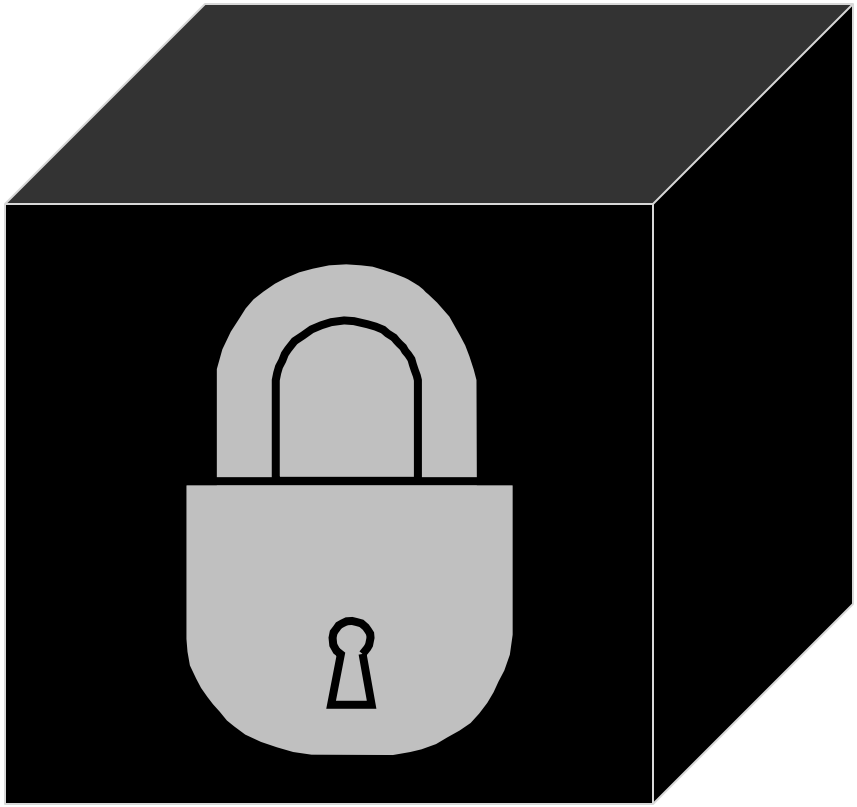

1-May
2-May
3-May
4-May
5-May
6-May
7-May
8-May
9-May
10-May
11-May
12-May
13-May
14-May
15-May
16-May
17-May
18-May
19-May
20-May
21-May
22-May
23-May
24-May
25-May
26-May
27-May
28-May
29-May
30-May


SPM

```
/*+
process_in_a_way_that_works_perfectly_today_under_the_current_environment
*/
```



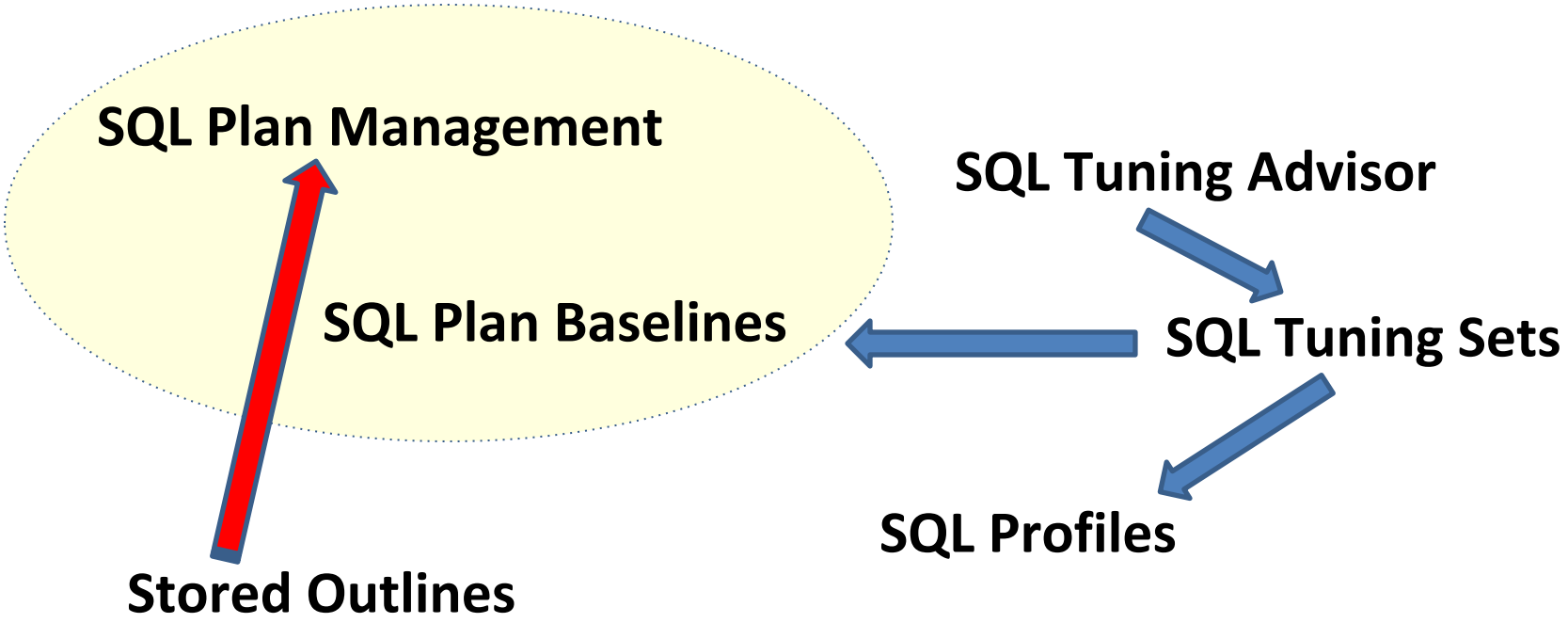
Code Change -> Full Regression Testing -> Approvals -> Release



Vendor Applications


SPM

Oracle's Name Game



SQL Plan Baselines vs. SQL Profiles

- **SQL Profiles =** Corrections for optimizer statistics
  - Requires Tuning pack
- **SQL Plan Baselines =** Set of accepted execution plans
  - Comes with Enterprise Edition 11g and higher

When to use each?

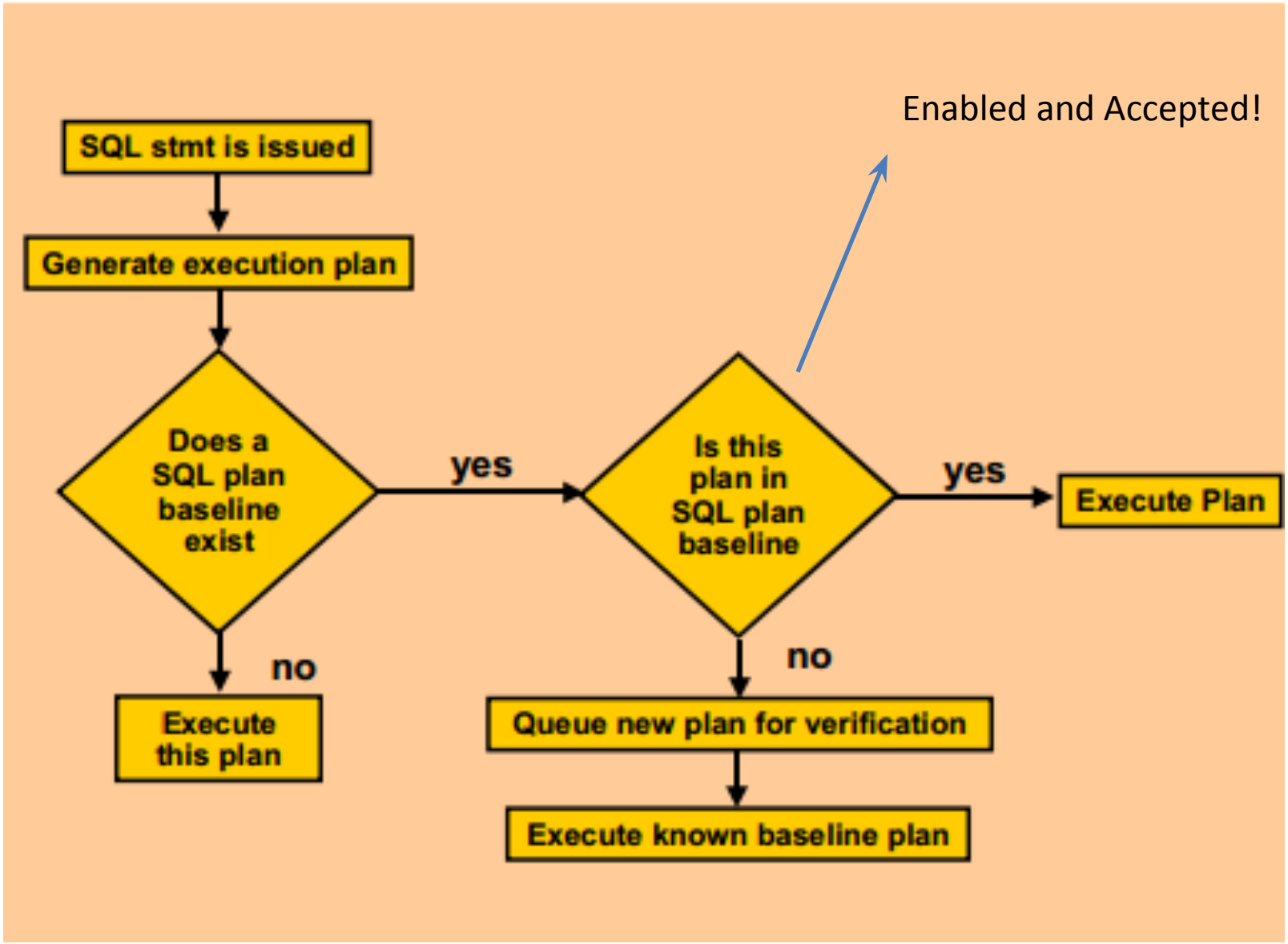
- **SQL Profiles =** Keep optimizer free to immediately adapt to changes (e.g. new stats)
- **SQL Plan Baselines =** Keep execution plans stable until you decide they should evolve

3 Parts to SPM

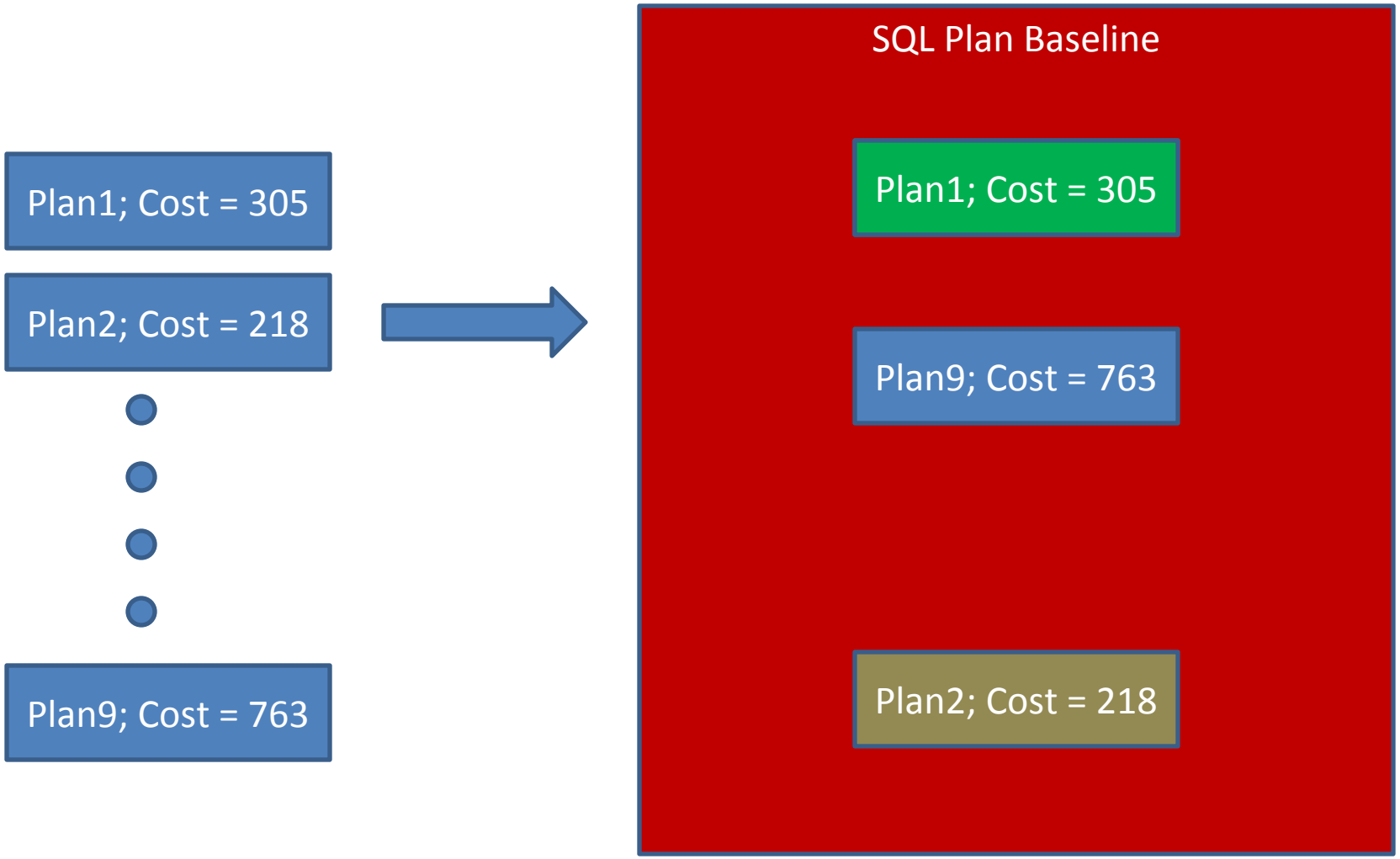
- Plan Capture
- Plan Selection
- Plan Evolution

Enable / Disable SPM

- optimizer_use_sql_plan_baselines
  - TRUE / FALSE
  - Default is TRUE (enabled)
  - No special license is required (just Enterprise Edition)


Enabled and Accepted!
SQL stmt is issued
Generate execution plan
Does a SQL plan baseline exist
yes
Is this plan in SQL plan baseline
yes
Execute Plan
no
Execute this plan
no
Queue new plan for verification
Execute known baseline plan

Choosing Execution Plan



Plays Nice With Others

- SQL Profiles
  - Still uses corrected statistics to calculate cost
- Adaptive Cursor Sharing
  - Still uses bind values for bind sensitive queries to choose best plan

Mostly Plays Nice With Others

- 12c Adaptive Query Optimization
  - Oracle can change execution plan as query being executed
    - For example, decision to use Hash Join vs. Nested Loops can be delayed until Oracle has better idea of how many rows returned from driving table
- Only final plan captured as baseline
  - If baseline enabled, adaptive process is skipped on subsequent executions

3 Parts to SPM

- Plan Capture 
- Plan Selection
- Plan Evolution

Plan Capture

- Automatic Capture
  - OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES to TRUE
    - Default is False
    - Dynamic
- Manual Load
  - From:
    - Cursor Cache
    - SQL Tuning Sets
    - Stored Outlines
    - Staging Table

Automatic Capture

- Stability of Execution Plans across system
  - Perfect Solution, Right?

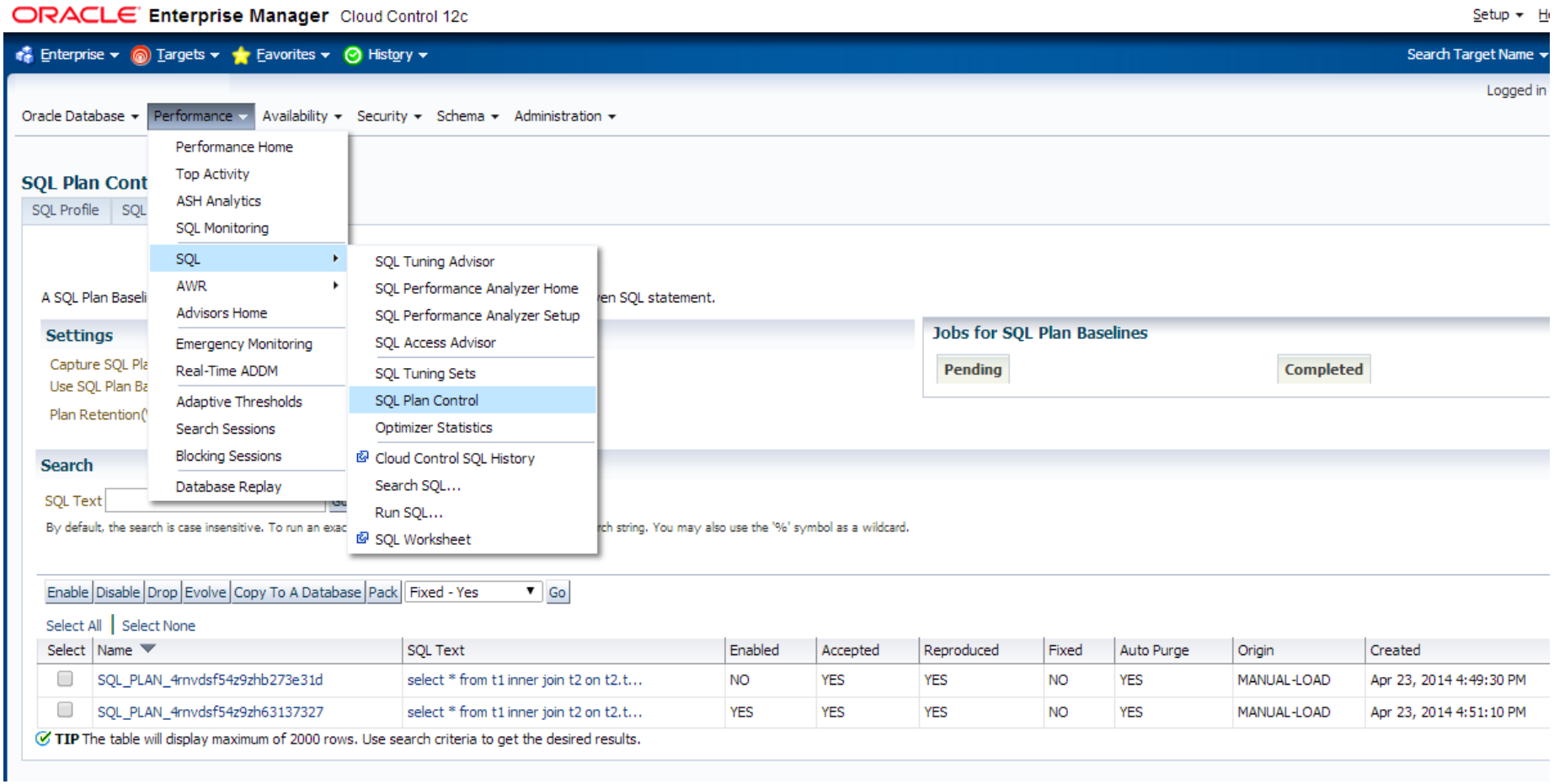
There is a reason the optimizer exists!!

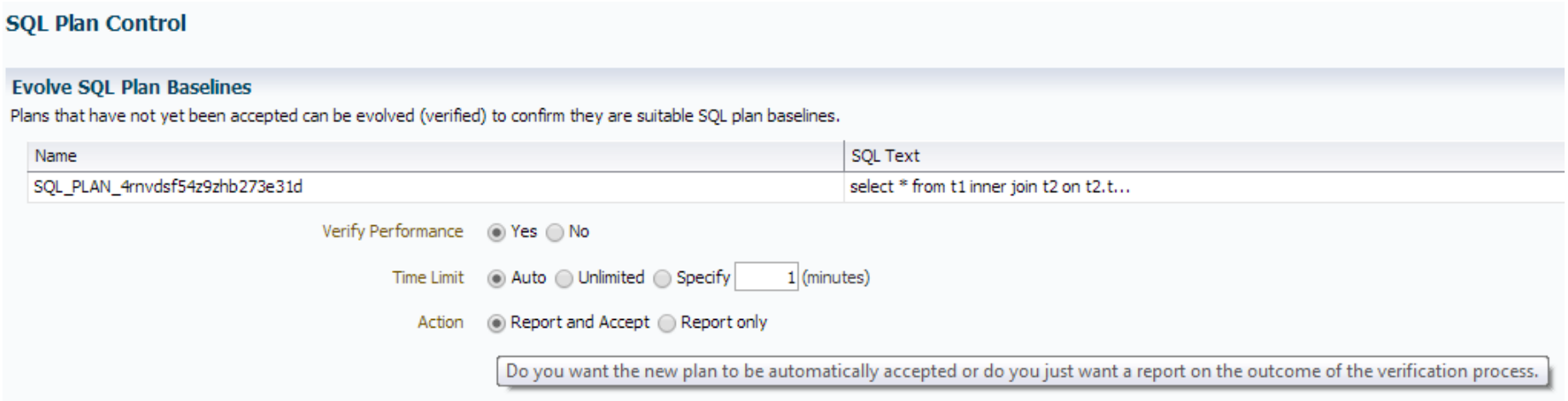
My Advice!

- Keep number of baselines to a manageable number
- Work to remove existing baselines
  - Figure out how to make optimizer come up with good plan on its own

3 Parts to SPM

- Plan Capture
- Plan Selection
- Plan Evolution


ORACLE Enterprise Manager Cloud Control 12c
Setup
Enterprise
Targets
Favorites
History
Search Target Name
Logged in
Oracle Database
Performance
Availability
Security
Schema
Administration
Performance Home
Top Activity
ASH Analytics
SQL Monitoring
SQL
AWR
Advisors Home
Emergency Monitoring
Real-Time ADDM
Adaptive Thresholds
Search Sessions
Blocking Sessions
Database Replay
SQL Tuning Advisor
SQL Performance Analyzer Home
SQL Performance Analyzer Setup
SQL Access Advisor
SQL Tuning Sets
SQL Plan Control
Optimizer Statistics
Cloud Control SQL History
Search SQL...
Run SQL...
SQL Worksheet
SQL Plan Cont
SQL Profile
Settings
Jobs for SQL Plan Baselines
Pending
Completed
Search
SQL Text
Enable
Disable
Drop
Evolve
Copy To A Database
Pack
Fixed - Yes
Go
Select All
Select None
Select
Name
SQL Text
Enabled
Accepted
Reproduced
Fixed
Auto Purge
Origin
Created
SQL_PLAN_4rnvdsf54z9zhb273e31d
select * from t1 inner join t2 on t2.t...
NO
YES
YES
NO
YES
MANUAL-LOAD
Apr 23, 2014 4:49:30 PM
SQL_PLAN_4rnvdsf54z9zh63137327
select * from t1 inner join t2 on t2.t...
YES
YES
YES
NO
YES
MANUAL-LOAD
Apr 23, 2014 4:51:10 PM
TIP The table will display maximum of 2000 rows. Use search criteria to get the desired results.


SQL Plan Control
Evolve SQL Plan Baselines
Plans that have not yet been accepted can be evolved (verified) to confirm they are suitable SQL plan baselines.
Name
SQL Text
SQL_PLAN_4rnvdsf54z9zhb273e31d
select * from t1 inner join t2 on t2.t...
Verify Performance
Yes
No
Time Limit
Auto
Unlimited
Specify
1
(minutes)
Action
Report and Accept
Report only
Do you want the new plan to be automatically accepted or do you just want a report on the outcome of the verification process.

Manually Evolve

```
declare
    v_report CLOB;
begin
    v_report := DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE(
                    sql_handle => 'SQL_4bd36dc38a4fa7f0',
                    plan_name => 'SQL_PLAN_4rnvdsf54z9zhc1cd3aac',
                    time_limit => DBMS_SPM.AUTO_LIMIT,
                    verify => 'YES',
                    commit => 'NO'  -- YES = Evolve, NO = Just generate report
                );
    dbms_output.put_line(v_report);
end;
```

```
------------------------------------------------------------------------------
                        Evolve SQL Plan Baseline Report
------------------------------------------------------------------------------

Inputs:
-------
  SQL_HANDLE = SQL_4bd36dc38a4fa7f0
  PLAN_NAME  = SQL_PLAN_4rnvdsf54z9zhc1cd3aac
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = NO

Plan: SQL_PLAN_4rnvdsf54z9zhc1cd3aac
------------------------------------
  Plan was verified: Time used .2 seconds.
  Plan failed performance criterion: performance equal to baseline plan.

                           Baseline Plan      Test Plan       Stats Ratio
                           -------------      ---------       -----------
  Execution Status:             COMPLETE       COMPLETE
  Rows Processed:                    300            300
  Elapsed Time(ms):                9.747          8.285              1.18
  CPU Time(ms):                    9.776          8.332              1.17
  Buffer Gets:                       400            400                 1
  Physical Read Requests:              0              0
  Physical Write Requests:             0              0
  Physical Read Bytes:                 0              0
  Physical Write Bytes:                0              0
  Executions:                          1              1

------------------------------------------------------------------------------
                                  Report Summary
------------------------------------------------------------------------------
Number of plans verified: 1
Number of plans accepted: 0
```

Automatic Plan Evolution

- 12c feature
- Nightly job (SYS_AUTO_SPM_EVOVLE_TASK)
  - All non-accepted plans ranked (newly found plans ranked highest) and evolve process run for as many as possible before maintenance window ends
  - Plans with better performance are accepted
- DBMS_SPM.SET_EVOLVE_TASK_PARAMETER
  - Can change settings of auto task
- Can be run manually:
  - DBMS_SPM.CREATE_EVOLVE_TASK
  - DBMS_SPM.EXECUTE_EVOLVE_TASK
  - DBMS_SPM.REPORT_EVOLVE_TASK
  - … and several more…

Actual Execution Plan Stored

- 12c feature
  - 11g only stores Outline
  - Allows for seeing execution plan even when it can no longer be reproduced

Troubleshooting

SPM Information can be gathered into trace files:

In 11gR1:

alter session set events 'trace [sql_planmanagement.*]';

In 11gR2 and up:

alter session set events 'trace [SQL_Plan_Management.*]';

Can also be seen in 10053 trace file:

```
begin
    dbms_sqldiag.dump_trace(
        p_sql_id => '5484zucv2dfgu',
        p_child_number => 1,
        p_component => 'Compiler',
        p_File_id => 'cm_test_spm_trc'
    );
end;
```

See http://structureddata.org/2011/08/18/creating-optimizer-trace-files/ for details

How do I personally use SPM?*

Automatic Workload Repository

- Stores snapshots of workload for historical diagnosis
- Requires Diagnostic Pack
- By default, interval = 1 hour / retention = 7 days
  - Can be increased
  - View current:

```
select snap_interval, retention
from dba_hist_wr_control;
```

```
-- AWR Details for single sql_id
select snap.begin_interval_time, ss.sql_id, ss.plan_hash_value
from
   dba_hist_sqlstat ss
   inner join dba_hist_snapshot snap
      on snap.snap_id = ss.snap_id
      and snap.instance_number = ss.instance_number
where ss.sql_id = '5484zucv2dfgu'
order by snap.begin_interval_time desc;
```

BEGIN_INTERVAL_TIME	SQL_ID	PLAN_HASH_VALUE
3/30/2014 10:00:34.747 AM	5484zucv2dfgu	1816303856
3/30/2014 9:00:32.569 AM	5484zucv2dfgu	1816303856
3/30/2014 8:00:30.454 AM	5484zucv2dfgu	3136000890
3/30/2014 7:00:28.308 AM	5484zucv2dfgu	3136000890
3/30/2014 6:00:26.116 AM	5484zucv2dfgu	3136000890
3/30/2014 5:00:23.986 AM	5484zucv2dfgu	3136000890

-- Get Plan for specific SQL_ID / Plan Hash Value

```
select *
from table(dbms_xplan.display_awr('5484zucv2dfgu','1816303856','','+OUTLINE'));
```

```
PLAN_TABLE_OUTPUT
SQL_ID 5484zucv2dfgu
--------------------
select * from t1     inner join t2         on t2.t2_id = t1.t2_id
inner join t3         on t3.t3_id = t1.t3_id     inner join t4
on t4.t4_id = t3.t4_id         and t4.t4_id = t2.t4_id where
t2.t2_id between 20 and 30

Plan hash value: 1816303856

-------------------------------------------------------------------------------------
| Id  | Operation                | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |      |       |       |    12 (100)|          |
|   1 |  HASH JOIN               |      |     1 |  2089 |    12   (9)| 00:00:01 |
|   2 |   HASH JOIN              |      |     1 |  2072 |     9  (12)| 00:00:01 |
|   3 |    MERGE JOIN CARTESIAN  |      |     1 |  2055 |     5   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL    | T1   |     1 |  2041 |     2   (0)| 00:00:01 |
|   5 |     BUFFER SORT          |      |    75 |  1050 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL   | T4   |    75 |  1050 |     3   (0)| 00:00:01 |
|   7 |    TABLE ACCESS FULL     | T2   |    12 |   204 |     3   (0)| 00:00:01 |
|   8 |   TABLE ACCESS FULL      | T3   |   100 |  1700 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('11.2.0.2')
      DB_VERSION('11.2.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$EE94F965")
      MERGE(@"SEL$9E43CB6E")
      OUTLINE(@"SEL$4")
      OUTLINE(@"SEL$9E43CB6E")
      MERGE(@"SEL$58A6D7F6")
      OUTLINE(@"SEL$3")
      OUTLINE(@"SEL$58A6D7F6")
      MERGE(@"SEL$1")
      OUTLINE(@"SEL$2")
      OUTLINE(@"SEL$1")
      FULL(@"SEL$EE94F965" "T1"@"SEL$1")
      FULL(@"SEL$EE94F965" "T4"@"SEL$3")
      FULL(@"SEL$EE94F965" "T2"@"SEL$1")
      FULL(@"SEL$EE94F965" "T3"@"SEL$2")
      LEADING(@"SEL$EE94F965" "T1"@"SEL$1" "T4"@"SEL$3" "T2"@"SEL$1"
              "T3"@"SEL$2")
      USE_MERGE_CARTESIAN(@"SEL$EE94F965" "T4"@"SEL$3")
      USE_HASH(@"SEL$EE94F965" "T2"@"SEL$1")
      USE_HASH(@"SEL$EE94F965" "T3"@"SEL$2")
      END_OUTLINE_DATA
  */
```

Outline => Able to reproduce exact same execution in Dev environments

```
-- Get actual SQL text
select sql_text
from dba_hist_sqltext
where sql_id = '5484zucv2dfgu';
```

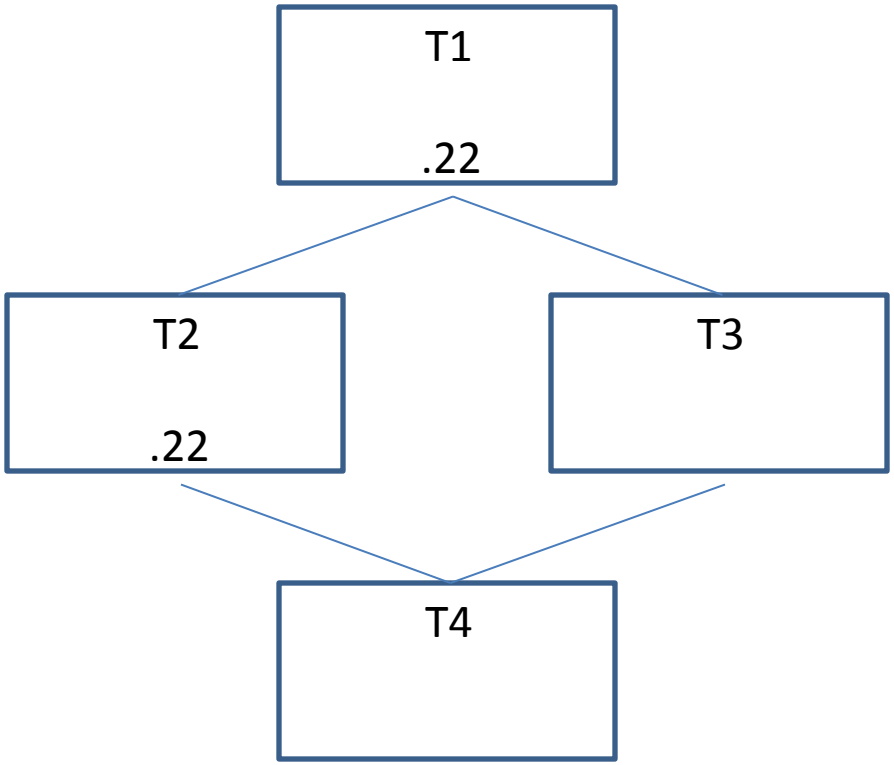
SQL Text => Able to get same SQL_ID in Dev environments

At this point I move to dev environment

Run on Dev, gathering extra information about execution plan

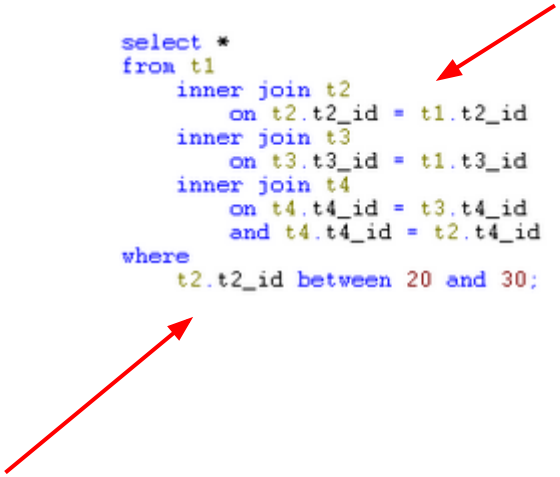
```
alter session set statistics_level = ALL;

select *
from t1
    inner join t2
        on t2.t2_id = t1.t2_id
    inner join t3
        on t3.t3_id = t1.t3_id
    inner join t4
        on t4.t4_id = t3.t4_id
        and t4.t4_id = t2.t4_id
where
    t2.t2_id between 20 and 30;

select *
from table(dbms_xplan.display_cursor('5484zucv2dfgu',1,'IOSTATS +OUTLINE'));

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('11.2.0.2')
      DB_VERSION('11.2.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$EE94F965")
      MERGE(@"SEL$9E43CB6E")
      OUTLINE(@"SEL$4")
      OUTLINE(@"SEL$9E43CB6E")
      MERGE(@"SEL$58A6D7F6")
      OUTLINE(@"SEL$3")
      OUTLINE(@"SEL$58A6D7F6")
      MERGE(@"SEL$1")
      OUTLINE(@"SEL$2")
      OUTLINE(@"SEL$1")
      FULL(@"SEL$EE94F965" "T1"@"SEL$1")
      FULL(@"SEL$EE94F965" "T4"@"SEL$3")
      FULL(@"SEL$EE94F965" "T2"@"SEL$1")
      FULL(@"SEL$EE94F965" "T3"@"SEL$2")
      LEADING(@"SEL$EE94F965" "T1"@"SEL$1" "T4"@"SEL$3" "T2"@"SEL$1" "T3"@"SEL$2")
      USE_MERGE_CARTESIAN(@"SEL$EE94F965" "T4"@"SEL$3")
      USE_HASH(@"SEL$EE94F965" "T2"@"SEL$1")
      USE_HASH(@"SEL$EE94F965" "T3"@"SEL$2")
      END_OUTLINE_DATA
  */

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

   1 - access("T4"."T4_ID"="T3"."T4_ID" AND "T3"."T3_ID"="T1"."T3_ID")
   2 - access("T4"."T4_ID"="T2"."T4_ID" AND "T2"."T2_ID"="T1"."T2_ID")
   4 - filter(("T1"."T2_ID"<=30 AND "T1"."T2_ID">=20))
   7 - filter(("T2"."T2_ID"<=30 AND "T2"."T2_ID">=20))
```

Visual SQL Tuning



Preferred join order: T2, T4, T3, T1

Add Indexes?



```
select *
from t1
    inner join t2
        on t2.t2_id = t1.t2_id
    inner join t3
        on t3.t3_id = t1.t3_id
    inner join t4
        on t4.t4_id = t3.t4_id
        and t4.t4_id = t2.t4_id
where
    t2.t2_id between 20 and 30;
```

t1 → 22% of rows match filter

What if filter was t2_id = 20? → 2% of rows match filter

data_selectivity.sql t1 t2_id 20

T2_ID	BLOCK_SELECTIVITY	BLOCK_COUNT	ROW_SELECTIVITY	ROW_COUNT
25	100	371	2	2000
30	100	371	2	2000
34	100	371	2	2000
1	100	371	2	2000
42	100	371	2	2000
22	100	371	2	2000
43	100	371	2	2000
44	100	371	2	2000
47	100	371	2	2000
13	100	371	2	2000
6	100	371	2	2000
28	100	371	2	2000
29	100	371	2	2000
11	100	371	2	2000
20	100	371	2	2000
21	100	371	2	2000
14	100	371	2	2000
26	100	371	2	2000
31	100	371	2	2000
2	100	371	2	2000

20 rows selected.

Use hints to get the execution plan you want

```
select /*+ cm2_test1 leading(t2 t4 t3 t1) */ *
from t1
    inner join t2
        on t2.t2_id = t1.t2_id
    inner join t3
        on t3.t3_id = t1.t3_id
    inner join t4
        on t4.t4_id = t3.t4_id
        and t4.t4_id = t2.t4_id
where
    t2.t2_id between 20 and 30;
```

Find new query SQL_ID in cursor cache

```
select vs.sql_id, vs.child_number, vs.*
from gv$sql vs
where last_active_time > sysdate - :minutes/24/60
and parsing_schema_name = :username
and sql_text like '%+ cm2_test1%'
order by last_active_time desc;
```

```
/*
    Check both SQL Statements are in Cursor Cache
*/
select sql_id, plan_hash_value, sql_text
from gv$sql
where sql_id in ('5484zucv2dfgu','abz14a2fpb5y8');
```

SQL_ID	PLAN_HASH_VALUE	SQL_TEXT
5484zucv2dfgu	1816303856	select * from t1 inner join t2 on t2
abz14a2fpb5y8	3136000890	select /*+ cm2_test1 leading(t2 t4 t3 t1) */ * fr

Current Situation

5484zucv2dfgu → 1816303856

abz14a2fpb5y8 → 3136000890

Desired Situation

5484zucv2dfgu → 1816303856

abz14a2fpb5y8 → 3136000890

```
/*
    Add bad plan manually to SQL PLAN BASELINE
        Note: this adds all plans for this SQL_ID currently in the cursor cache.
            If you want to just add one, there is another parameter plan_hash_value
            that can be specified.
*/
DECLARE
    v_cnt PLS_INTEGER;
BEGIN
    v_cnt := dbms_spm.load_plans_from_cursor_cache(sql_id=>'5484zucv2dfgu');
END;
```

Function, returns how many plans were loaded

```
/*
    Check that plan loaded successfully, get sql_handle and plan_name for next step
*/
select to_char(signature) signature, sql_handle, plan_name, enabled, accepted, to_char(sql_text) sql_text
from dba_sql_plan_baselines
where created > sysdate - 15/24/60
order by created desc;
```

SIGNATURE	SQL_HANDLE	PLAN_NAME	ENABLED	ACCEPTED	SQL_TEXT
5463831459537070064	SQL_4bd36dc38a4fa7f0	SQL_PLAN_4rnvdsf54z9zhb273e31d	YES	YES	select *from t1 inner join t2

```
/*
    Disable old plan.  If multiple plans were loaded, this step needs to be done
        once for each one.
*/
DECLARE
    v_cnt PLS_INTEGER;
BEGIN
    v_cnt := DBMS_SPM.ALTER_SQL_PLAN_BASELINE(
         sql_handle =>      'SQL_4bd36dc38a4fa7f0',             -- sql_handle from dba_sql_plan_baselines
         plan_name  =>      'SQL_PLAN_4rnvdsf54z9zhb273e31d',   -- plan_name from dba_sql_plan_baselines
         attribute_name =>  'enabled',
         attribute_value => 'NO');
END;


/*
     Check plans are disabled
*/
select enabled, plan_name
from dba_sql_plan_baselines
where sql_handle = 'SQL_4bd36dc38a4fa7f0';
```

ENABLED	PLAN_NAME
NO	SQL_PLAN_4rnvdsf54z9zhb273e31d

Current Situation

5484zucv2dfgu

1816303856

abz14a2fpb5y8

3136000890

```
/*
    Reassign good plan to old SQL
*/
DECLARE
   v_cnt PLS_INTEGER;
BEGIN
    v_cnt := dbms_spm.load_plans_from_cursor_cache (
                sql_id => 'abz14a2fpb5y8',              -- new SQL
                plan_hash_value => '3136000890',        -- new Plan
                sql_handle => 'SQL_4bd36dc38a4fa7f0'    -- SQL handle for poorly performing query
             );
END;
```

```
-- Check new details
select sql_handle, plan_name, enabled
from dba_sql_plan_baselines
where sql_handle = 'SQL_4bd36dc38a4fa7f0';
```

SQL_HANDLE	PLAN_NAME	ENABLED
SQL_4bd36dc38a4fa7f0	SQL_PLAN_4rnvdsf54z9zh63137327	YES
SQL_4bd36dc38a4fa7f0	SQL_PLAN_4rnvdsf54z9zhb273e31d	NO

Current Situation

5484zucv2dfgu

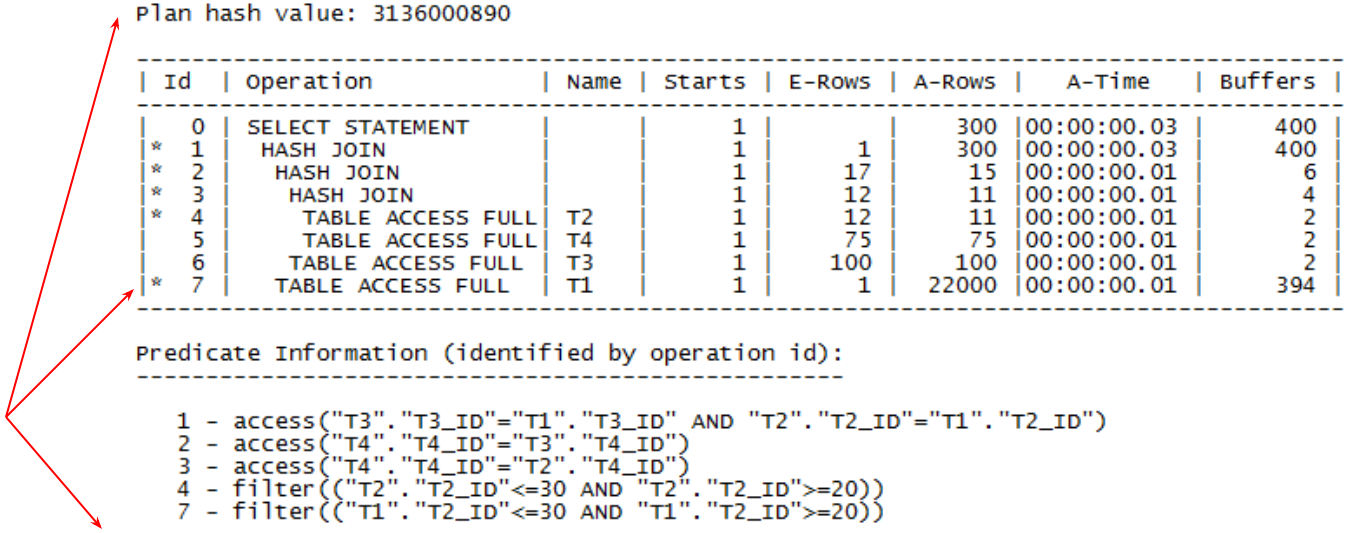
abz14a2fpb5y8

1816303856

3136000890

Lets check!

```
/*=======================================

    Check to ensure SPM working

    Note:  You will need to invalidate any old
        plans from the Cursor Cache.  If SPM is being used
        there will be a comment in the Notes section of DBMS_XPLAN.DISPLAY_CURSOR
        as well as the column SQL_PLAN_BASELINE will be populated in gv$sql

=======================================*/

alter session set statistics_level = ALL

select *
from t1
    inner join t2
        on t2.t2_id = t1.t2_id
    inner join t3
        on t3.t3_id = t1.t3_id
    inner join t4
        on t4.t4_id = t3.t4_id
        and t4.t4_id = t2.t4_id
where
    t2.t2_id between 20 and 30;

select *
from table(dbms_xplan.display_cursor('5484zucv2dfgu',1,'IOSTATS'));
```



```
Plan hash value: 3136000890

-------------------------------------------------------------------------------------------
| Id  | Operation            | Name | Starts | E-Rows | A-Rows |   A-Time   | Buffers |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT     |      |      1 |        |    300 |00:00:00.03 |     400 |
|*  1 |  HASH JOIN           |      |      1 |      1 |    300 |00:00:00.03 |     400 |
|*  2 |   HASH JOIN          |      |      1 |     17 |     15 |00:00:00.01 |       6 |
|*  3 |    HASH JOIN         |      |      1 |     12 |     11 |00:00:00.01 |       4 |
|*  4 |     TABLE ACCESS FULL| T2   |      1 |     12 |     11 |00:00:00.01 |       2 |
|   5 |     TABLE ACCESS FULL| T4   |      1 |     75 |     75 |00:00:00.01 |       2 |
|   6 |    TABLE ACCESS FULL | T3   |      1 |    100 |    100 |00:00:00.01 |       2 |
|*  7 |   TABLE ACCESS FULL  | T1   |      1 |      1 |  22000 |00:00:00.01 |     394 |
-------------------------------------------------------------------------------------------

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

   1 - access("T3"."T3_ID"="T1"."T3_ID" AND "T2"."T2_ID"="T1"."T2_ID")
   2 - access("T4"."T4_ID"="T3"."T4_ID")
   3 - access("T4"."T4_ID"="T2"."T4_ID")
   4 - filter(("T2"."T2_ID"<=30 AND "T2"."T2_ID">=20))
   7 - filter(("T1"."T2_ID"<=30 AND "T1"."T2_ID">=20))

Note
-----
   - SQL plan baseline SQL_PLAN_4rnvdsf54z9zh63137327 used for this statement
```

We can also see SPM being used via GV$SQL

```
select sql_id, child_number, plan_hash_value, sql_plan_baseline, sql_text
from gv$sql
where sql_id = '5484zucv2dfgu';
```

SQL_ID	CHILD_NUMBER	PLAN_HASH_VALUE	SQL_PLAN_BASELINE	SQL_TEXT
5484zucv2dfgu	1	3136000890	SQL_PLAN_4rnvdsf54z9zh63137327	select * from t1 inner join t2

Copy new SPM back to Prod

On DEV:

```
/*
    Create staging table for holding plans to transfer
*/

begin
    DBMS_SPM.CREATE_STGTAB_BASELINE (
        table_name => :stg_table_name
    );
end;



/*
    Put plans specified by SQL Handle (from DBA_SQL_PLAN_BASELINES)
    into staging table
*/

declare
    v_result NUMBER;
begin
    v_result := DBMS_SPM.PACK_STGTAB_BASELINE(
        table_name => :stg_table_name,
        sql_handle => :spm_sql_handle
    );
    dbms_output.put_line('Number of Plans packed: '||v_result);
end;
```

Datapump Export of Staging Table

Copy new SPM back to Prod

On Prod:

Datapump Import of Staging Table

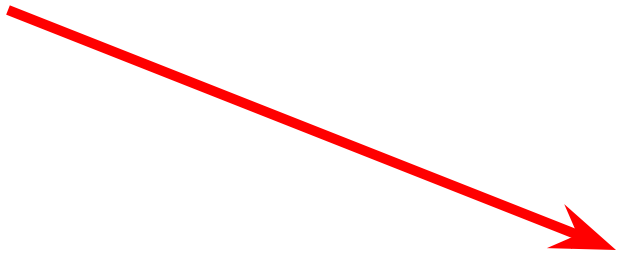
```
/*
    Load plans from staging table
*/

declare
    v_result NUMBER;
begin
    v_result := DBMS_SPM.UNPACK_STGTAB_BASELINE(
        table_name => :stg_table_name,
        sql_handle => :spm_sql_handle
    );
    dbms_output.put_line('Number of Plans unpacked: '||v_result);
end;
```

Check results!

You may need to flush cursor from cursor cache for new plans to take effect:
DBMS_SHARED_POOL.PURGE ('**ADDRESS, HASH_VALUE**', '**C**');

Relax!


SPM



How do I personally use SPM?*

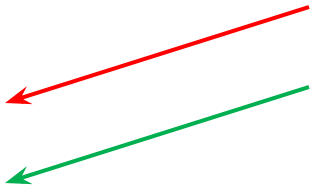
Automatic Workload Repository

- Stores snapshots of workload for historical diagnosis
- Requires Diagnostic Pack
- By default, interval = 1 hour / retention = 7 days
  - Can be increased
  - View current:

```
select snap_interval, retention
from dba_hist_wr_control;
```

```
-- AWR Details for single sql_id
select snap.begin_interval_time, ss.sql_id, ss.plan_hash_value
from
   dba_hist_sqlstat ss
   inner join dba_hist_snapshot snap
      on snap.snap_id = ss.snap_id
      and snap.instance_number = ss.instance_number
where ss.sql_id = '5484zucv2dfgu'
order by snap.begin_interval_time desc;
```

BEGIN_INTERVAL_TIME	SQL_ID	PLAN_HASH_VALUE
3/30/2014 10:00:34.747 AM	5484zucv2dfgu	1816303856
3/30/2014 9:00:32.569 AM	5484zucv2dfgu	1816303856
3/30/2014 8:00:30.454 AM	5484zucv2dfgu	3136000890
3/30/2014 7:00:28.308 AM	5484zucv2dfgu	3136000890
3/30/2014 6:00:26.116 AM	5484zucv2dfgu	3136000890
3/30/2014 5:00:23.986 AM	5484zucv2dfgu	3136000890



-- Get Plan for specific SQL_ID / Plan Hash Value

```
select *
from table(dbms_xplan.display_awr('5484zucv2dfgu','1816303856','','+OUTLINE'));
```

```
PLAN_TABLE_OUTPUT
SQL_ID 5484zucv2dfgu
--------------------
select * from t1     inner join t2         on t2.t2_id = t1.t2_id
inner join t3         on t3.t3_id = t1.t3_id     inner join t4
on t4.t4_id = t3.t4_id         and t4.t4_id = t2.t4_id where
t2.t2_id between 20 and 30

Plan hash value: 1816303856

-------------------------------------------------------------------------------------
| Id  | Operation                | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT         |      |       |       |    12 (100)|          |
|   1 |  HASH JOIN               |      |     1 |  2089 |    12   (9)| 00:00:01 |
|   2 |   HASH JOIN              |      |     1 |  2072 |     9  (12)| 00:00:01 |
|   3 |    MERGE JOIN CARTESIAN  |      |     1 |  2055 |     5   (0)| 00:00:01 |
|   4 |     TABLE ACCESS FULL    | T1   |     1 |  2041 |     2   (0)| 00:00:01 |
|   5 |     BUFFER SORT          |      |    75 |  1050 |     3   (0)| 00:00:01 |
|   6 |      TABLE ACCESS FULL   | T4   |    75 |  1050 |     3   (0)| 00:00:01 |
|   7 |    TABLE ACCESS FULL     | T2   |    12 |   204 |     3   (0)| 00:00:01 |
|   8 |   TABLE ACCESS FULL      | T3   |   100 |  1700 |     3   (0)| 00:00:01 |
-------------------------------------------------------------------------------------

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('11.2.0.2')
      DB_VERSION('11.2.0.2')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$EE94F965")
      MERGE(@"SEL$9E43CB6E")
      OUTLINE(@"SEL$4")
      OUTLINE(@"SEL$9E43CB6E")
      MERGE(@"SEL$58A6D7F6")
      OUTLINE(@"SEL$3")
      OUTLINE(@"SEL$58A6D7F6")
      MERGE(@"SEL$1")
      OUTLINE(@"SEL$2")
      OUTLINE(@"SEL$1")
      FULL(@"SEL$EE94F965" "T1"@"SEL$1")
      FULL(@"SEL$EE94F965" "T4"@"SEL$3")
      FULL(@"SEL$EE94F965" "T2"@"SEL$1")
      FULL(@"SEL$EE94F965" "T3"@"SEL$2")
      LEADING(@"SEL$EE94F965" "T1"@"SEL$1" "T4"@"SEL$3" "T2"@"SEL$1"
              "T3"@"SEL$2")
      USE_MERGE_CARTESIAN(@"SEL$EE94F965" "T4"@"SEL$3")
      USE_HASH(@"SEL$EE94F965" "T2"@"SEL$1")
      USE_HASH(@"SEL$EE94F965" "T3"@"SEL$2")
      END_OUTLINE_DATA
  */
```

Outline => Able to reproduce exact same execution in Dev environments

Extremely Quick!

- If good plan in AWR, typically takes < 15 minutes
  - Includes testing on Dev!

Resources

- Oracle Whitepaper : SQL Plan Management with Oracle Database 12c
- Oracle Optimizer Blog: https://blogs.oracle.com/optimizer
- Oracle Database Performance Tuning Guide
- Oracle Database PL/SQL Packages and Types Reference

Questions?