Controlling Execution Plans

(without touching the code)

Because there are just some things that no one wants to touch!

by Kerry Osborne

- an oldish Oracle guy

whoami

Started working with Oracle in 1983
Primarily a developer for the first 5-6 years
Became a consultant around 1987
(through no fault of my own)
…

Never worked directly for Oracle
Not certified in anything (except Scuba Diving)
But I have attended the Hotsos Symposium 5 years in a row!

It's the Code!

"*Let me blurt out the punch line of this article in one sentence. The main performance problem in the huge majority of database applications is bad SQL code.*"

Joe Celko - *It's the Code Stupid!*
Information Management Magazine, June 2005

Why is there so much "bad" code?

SQL is a very very very high level language

- Actually it's closer to a software spec than a program
  - Basically only the result is defined (I'm stretching here)
  - But many many implementation decisions are left to the DB
    - the most import input is the statistics
    - lots of optimizer parameters as well
      - 342 in 11.2.0.1 on linux
      - 2399 if you count the hidden ones
  - It can be like giving instructions to my kids

enkitec

Why is there so much “bad” code?

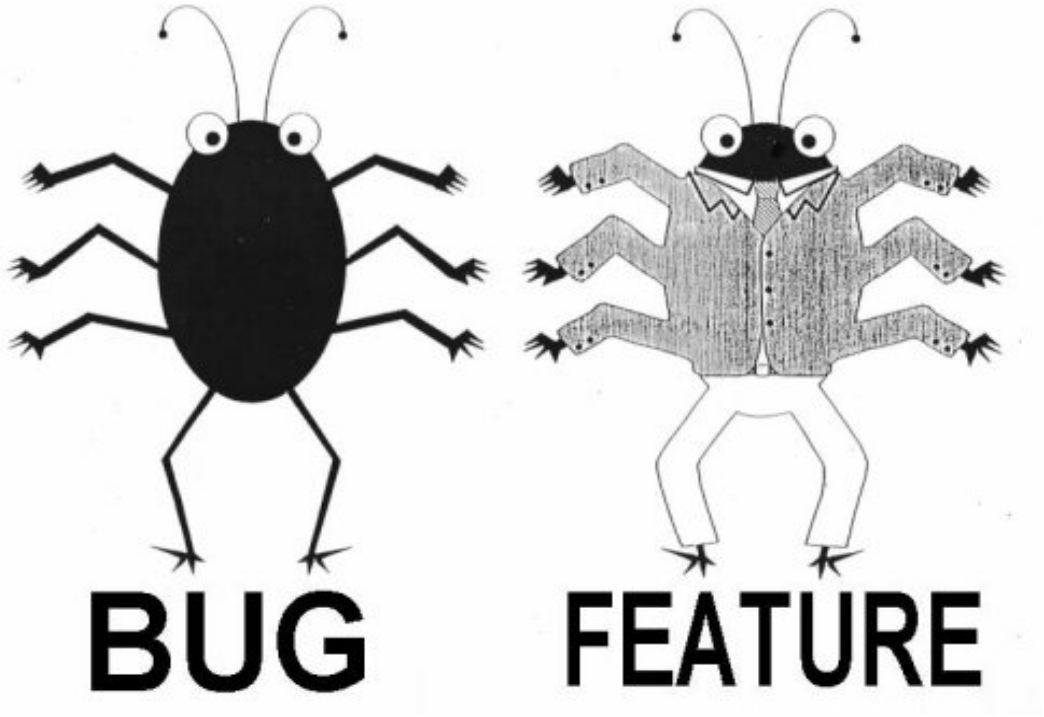
enkitec

So why can't we just "fix" the code?

- Sometimes it's Not Ours to Fix (i.e. packaged application)
- Sometimes there's Not Enough Time
  - it's an emergency
  - onerous change control (adding to_date function)
- Sometimes it's Not the Code!

Digression – Bind Variable Peeking

Drives Me Nuts!



enkitec

So What Can We Do?

enkitec

Some Possible Solutions?

Change Database Parameters (Big Knob Tuning)
Add additional access paths (Indexes)
Remove some access paths
Monkey with Stats

problem with these approaches –

they are very non-specific

Or We Can Use Hints Behind the Scenes

As of 11g there are 3 options

- Outlines (aka Plan Stability)
- SQL Profiles (SQL Tuning Advisor)
- SQL Baselines (SQL Plan Management)

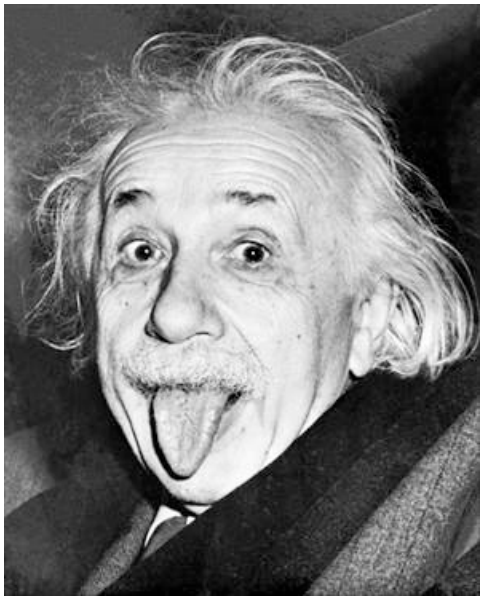
Each was created with a Different Goal
But they all work basically the same way
They each apply a set of hints behind the scenes
Each iteration has added something new to the mix

enkitec

Where Hint Based Mechanisms Work Well

A Few Statements with "Bad" plans

- **Plan Instability (bind variable peeking)**
- **Fixing optimizer shortcomings (correlated columns)**
- **Band Aids**



Note that they can have laser like specificity.

(I know it's a big word!)

Where They Don't Work Well

Anywhere there are lot's of problems

- Lot's of statements that have "Bad" plans
- Systemic Problems

Anywhere that the structure of a query needs to change

- Unions that should have been joins …
- Sub-queries (subquery factoring for example) …

```
Select col1 from skew where col2 = 'D'
Union all
Select col1 from skew where col12 = 'E'
Union all
Select col1 from skew where col12 = 'U';
```

```
Select col1 from kso.skew
where col4 in ('D' , 'E', 'Y');
```

Stored Outlines

Half Baked

- Goal was to “lock” plans
- Not enabled in any version by default
- Requires setting use_stored_outlines=true
- Sadly use_stored_outlines is not a real parameter
- Requires database trigger to enable them on startup
- Invalid hints are silently ignored
- There was an editor for a brief period
- Normalizes SQL text by collapsing spaces
- 10g added DBMS_OUTLN.CREATE_OUTLINE procedure
- Still works in 11g – but “deprecated”
- Overrides (disables) SQL Profiles and Baselines
- Still uses hash_value instead of sql_id
- Uses Categories (DEFAULT)

SQL Profiles

¾ Baked

- Goal was to apply statistical fixes
- Created by SQL Tuning Advisor (dbms_sqltune)
- Using semi-undocumented OPT_ESTIMATE hint
- Enabled by default
- * Can apply to multiple statements (force_matching)
- Invalid hints silently ignored
- Stored in SMB like SQL BASELINES (in 11g)
- * Provides procedure to import hints (import_sql_profile)
- Capable of applying any valid hints (I think)
- Uses Categories (DEFAULT)

SQL Tuning Advisor (STA) Profiles

So, a SQL profile is sort of like gathering statistics on A QUERY - which involves many tables, columns and the like....

In fact - it is just like gathering statistics for a query, it stores additional information in the dictionary which the optimizer uses at optimization time to determine the correct plan. The SQL Profile is not "locking a plan in place", but rather giving the optimizer yet more bits of information it can use to get the right plan.

~ Tom Kyte

OPT_ESTIMATE Hint

Applies Fudge Factors

- basically scales an optimizer calculation (up or down)
- valid (though undocumented) hint

```
OPT_ESTIMATE(@"SEL$5DA710D3", INDEX_FILTER, "F"@"SEL$1", IDX$$_1AA260002, SCALE_ROWS=8.883203639e-06)
OPT_ESTIMATE(@"SEL$5DA710D3", INDEX_SKIP_SCAN, "F"@"SEL$1", IDX$$_1AA260002, SCALE_ROWS=8.883203639e-06)
OPT_ESTIMATE(@"SEL$5DA710D3", JOIN, ("B"@"SEL$1", "A"@"SEL$1"), SCALE_ROWS=4.446153275)
OPT_ESTIMATE(@"SEL$5DA710D3", JOIN, ("C"@"SEL$1", "A"@"SEL$1"), SCALE_ROWS=7.884506683)
OPT_ESTIMATE(@"SEL$5DA710D3", JOIN, ("E"@"SEL$1", "A"@"SEL$1"), SCALE_ROWS=25.60960842)
OPT_ESTIMATE(@"SEL$5DA710D3", JOIN, ("F"@"SEL$1", "B"@"SEL$1"), SCALE_ROWS=26.34181566)
OPT_ESTIMATE(@"SEL$5DA710D3", JOIN, ("F"@"SEL$1", "B"@"SEL$1", "A"@"SEL$1"), SCALE_ROWS=839.9683673)
OPT_ESTIMATE(@"SEL$5DA710D3", TABLE, "D"@"SEL$1", SCALE_ROWS=5.083144565e+11)
OPT_ESTIMATE(@"SEL$5", INDEX_SCAN, "C"@"SEL$5", ORDER_FG_ITEM_IX3, SCALE_ROWS=0.2507281101)
```

HINT	SUBTYPE	COUNT(*)
OPT_ESTIMATE	INDEX_FILTER	12
OPT_ESTIMATE	INDEX_SCAN	32
OPT_ESTIMATE	INDEX_SKIP_SCAN	23
OPT_ESTIMATE	JOIN	154
OPT_ESTIMATE	TABLE	29

enkitec

STA Profiles (with OPT_ESTIMATE)

Goal appears to be applying statistical fix
Primarily using semi-undocumented OPT

I am really not a big fan, because …
1. They tend to "sour" over time

But they have a few redeeming qualities .
1. Good for indicating where the optimiz
2. Good for finding new plans (which ca
3. Maybe good for optimizer shortcomin

But …
They tend to "sour" over time!

Other STA Profile Hints

```
SQL> @sql_profile_distinct_hints
Enter value for profile_name: SYS_SQLPROF%

HINT                                                COUNT(*)
-------------------------------------------------- ----------
COLUMN_STATS                                               13
FIRST_ROWS                                                  1
IGNORE_OPTIM_EMBEDDED_HINTS                                 1
INDEX_STATS                                                 1
OPTIMIZER_FEATURES_ENABLE                                  14
OPT_ESTIMATE                                              178
TABLE_STATS                                                 2



SYS@LAB112> @sql_profile_hints
Enter value for profile_name: SYS_SQLPROF_0126f1743c7d0005

HINT
--------------------------------------------------------------------------------------------------------------
COLUMN_STATS("KSO"."SKEW", "PK_COL", scale, length=5)
COLUMN_STATS("KSO"."SKEW", "COL1", scale, length=4 distinct=828841 nulls=12.8723033 min=1 max=1000000)
TABLE_STATS("KSO"."SKEW", scale, blocks=162294 rows=35183107.66)
OPTIMIZER_FEATURES_ENABLE(default)
```

IMPORT_SQL_PROFILE

```
10.2 definition:

PROCEDURE IMPORT_SQL_PROFILE
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 SQL_TEXT                       CLOB                    IN
 PROFILE                        SQLPROF_ATTR            IN
 NAME                           VARCHAR2                IN     DEFAULT
 DESCRIPTION                    VARCHAR2                IN     DEFAULT
 CATEGORY                       VARCHAR2                IN     DEFAULT
 VALIDATE                       BOOLEAN                 IN     DEFAULT
 REPLACE                        BOOLEAN                 IN     DEFAULT
 FORCE_MATCH                    BOOLEAN                 IN     DEFAULT


SQL> desc sqlprof_attr
 sqlprof_attr VARRAY(2000) OF VARCHAR2(500)
```

Note: 11g overloads with PROFILE_XML which expects hints in a clob in XML format (exactly as they are stored in v$sql.other_xml).

SQL Baselines

Fully Baked (almost)

- Goal was to prevent performance regression
- (Closer to Outlines than to SQL Profiles)
- Enabled by default in 11g (optimizer_use_sql_plan_baselines)
- Capable of applying any valid hints
- * Has associated plan_hash_value
- Invalid hints are NOT silently ignored!
- Provides procedure to import plans (DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE)
- Overridden by Outlines
- Can work with SQL Profiles (merges hints)
- Can have multiple Baselines per statement
- No Categories
- Preferred Set (fixed=yes)

SQL Plan Management

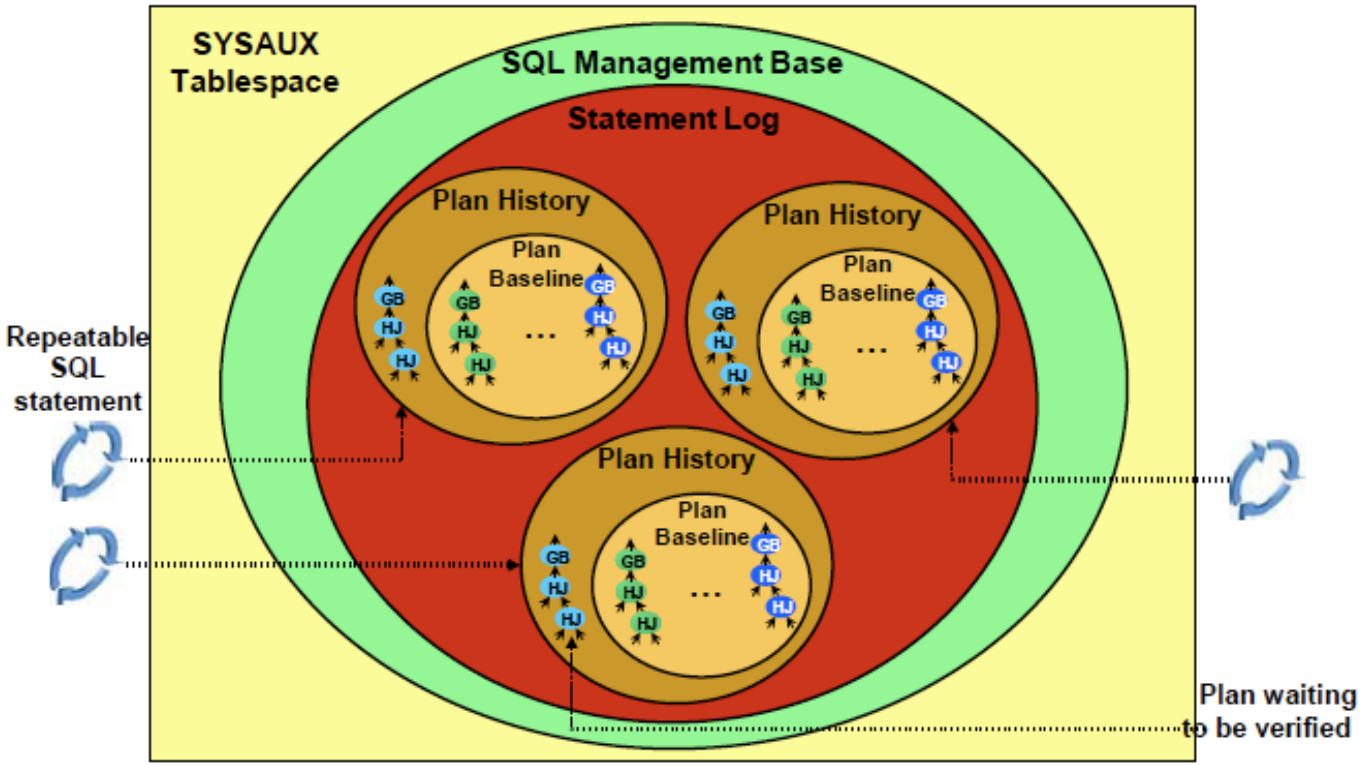
The Idea is to Prevent Backward Movement
New Framework using Baselines

SPM is On by default (sort of)

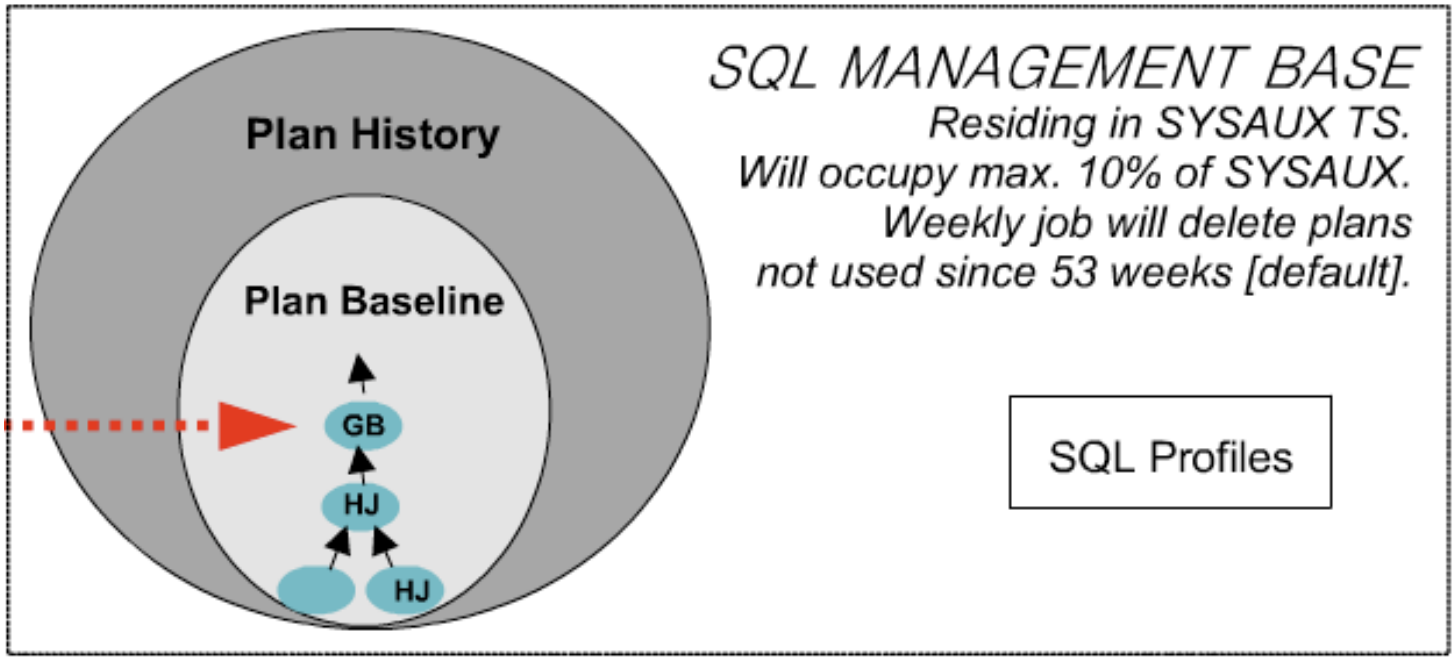
optimizer_use_sql_plan_baselines=true
But no plans are Baselined by default
Baselines can be bulk loaded
- From a SQL Tuning Set (10g)
- From Outlines
- From the cursor cache
- Via optimizer_capture_sql_plan_baselines=true

enkitec

SQL Plan Management

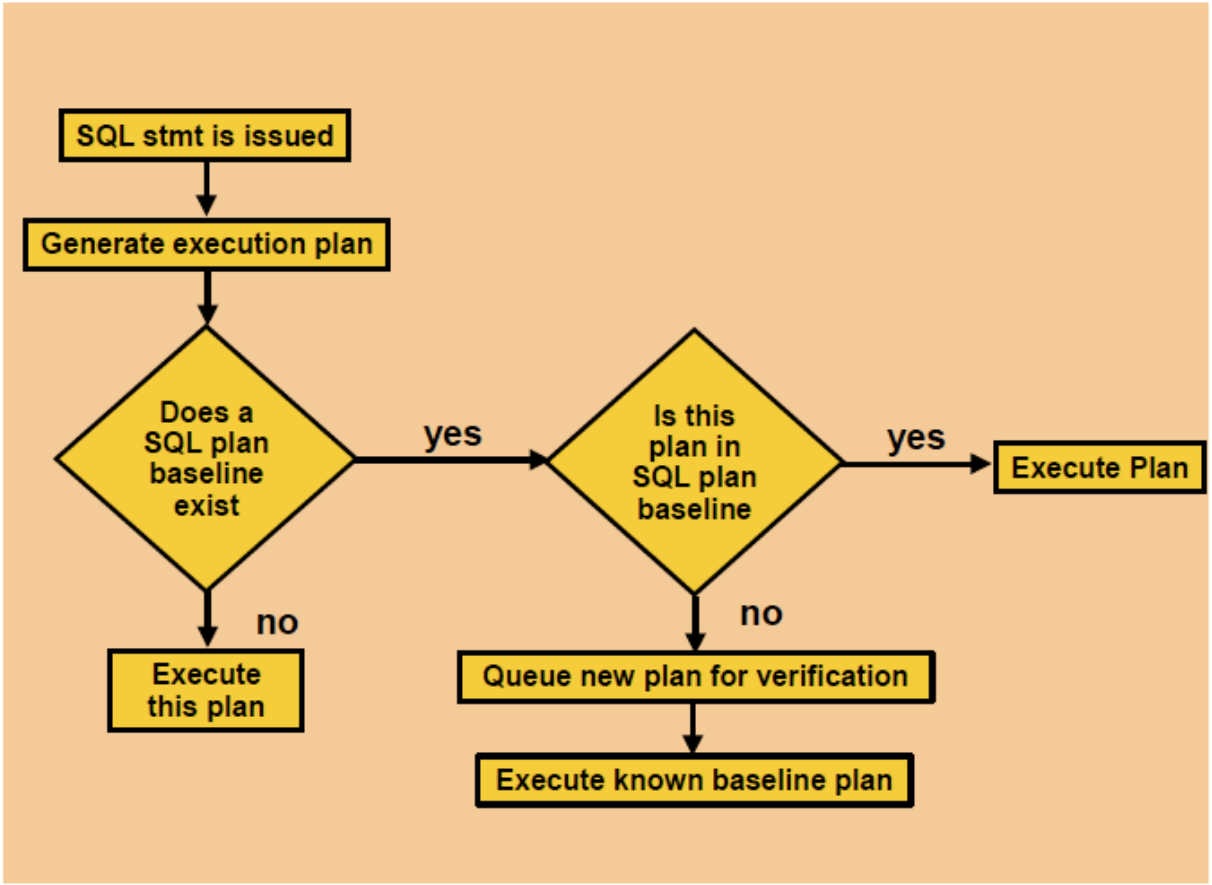


SQL Plan Management



Change defaults with DBMS_SPM.CONFIGURE (50% of SYSAUX is max)

SQL Plan Management – Hard Parse



SQL Plan Management

So what's actually stored?

- A plan hash value (calculated differently than v$sql)
- Hints to reproduce the plan
- Signature (no sql_id)
- The actual plan is not stored
- More complicated naming

```
SYS@LAB111> select spb.sql_handle, spb.plan_name, spb.sql_text,
  2  spb.enabled, spb.accepted, spb.fixed,
  3  to_char(spb.last_executed,'dd-mon-yy HH24:MI') last_executed
  4  from
  5  dba_sql_plan_baselines spb;

SQL_HANDLE                PLAN_NAME                      SQL_TEXT             ENABLED ACC FIX LAST_EXECUTED
------------------------- ------------------------------ -------------------- ------- --- --- ---------------
SYS_SQL_36bf1c88f777e894  SYS_SQL_PLAN_f777e89455381d08  select avg(pk_col) f YES     YES NO  27-oct-09 10:20
SYS_SQL_f2784d83c1974f5e  SYS_SQL_PLAN_c1974f5e54680e33  select avg(pk_col) f YES     YES NO  27-oct-09 11:12
SYS_SQL_f2784d83c1974f5e  SYS_SQL_PLAN_c1974f5e55381d08  select avg(pk_col) f YES     NO  NO
…
```

enkitec

So Which Is Most Useful?

And the Survey Says:

Profiles – No. 1 Answer
Baselines – No. 2 Answer

Why?

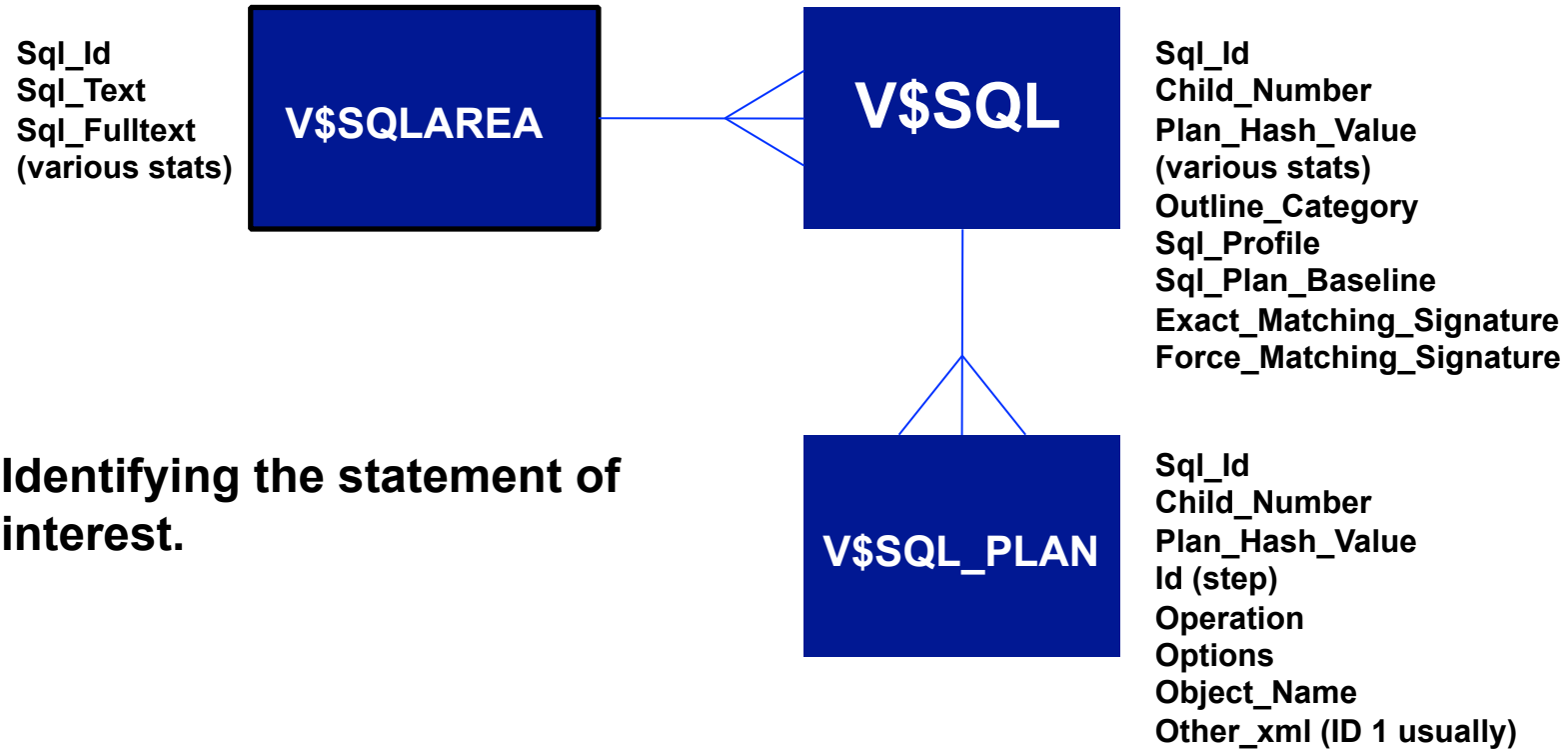
Profiles

dbms_sqltune.import_sql_profile
force_matching

Baselines

plan_hash_value
***no procedure to import hints**
***no force_matching**

Shared Pool Layout (V$SQL…)



Identifying the statement of interest.

Note: prior to 10g hash_value used as key (no sql_id)

Finding Statements in the Shared Pool

```
SQL> !cat find_sql.sql
select sql_id, child_number, plan_hash_value plan_hash, executions execs,
(elapsed_time/1000000)/decode(nvl(executions,0),0,1,executions) avg_etime,
disk_reads/decode(nvl(executions,0),0,1,executions) avg_pio,
buffer_gets/decode(nvl(executions,0),0,1,executions) avg_lio,
sql_text
from v$sql s
where upper(sql_text) like upper(nvl('&sql_text',sql_text))
and sql_text not like '%from v$sql where sql_text like nvl(%'
and sql_id like nvl('&sql_id',sql_id)
order by 1, 2, 3
/


SQL> @find_sql
Enter value for sql_text: %skew%
Enter value for sql_id:

SQL_ID         CHILD  PLAN_HASH  EXECS  AVG_ETIME      AVG_LIO SQL_TEXT
------------- ------ ---------- ------ ---------- ------------ ------------------------------------------------
0qa98gcnnza7h      0  568322376      5      13.09      142,646 select avg(pk_col) from kso.skew where col1 > 0
0qa98gcnnza7h      1 3723858078      1       9.80    2,626,102 select avg(pk_col) from kso.skew where col1 > 0
```

Explain Plan - Lies

```
SQL> explain plan for select …
SQL> select * from table(dbms_xplan.display('plan_table','','ALL'));
```

It tells you what it thinks the optimizer might do …
assuming the environment is the same as production
assuming that bind variable peeking doesn't come into play
etc…

(note: autotrace uses explain plan too)

The best liar is one that tells the truth most of the time.

Google for "Explain Plan Lies" for more info

Finding Plans for Statements in the Shared Pool

```
SQL> !cat dplan.sql
set lines 150
select * from table(dbms_xplan.display_cursor('&sql_id','&child_no','typical'))
/

SQL> @dplan
Enter value for sql_id: 0qa98gcnnza7h
Enter value for child_no: 0

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  0qa98gcnnza7h, child number 0
-------------------------------------
select avg(pk_col) from kso.skew where col1 > 0

Plan hash value: 568322376

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       | 31719 (100)|          |
|   1 |  SORT AGGREGATE    |      |     1 |    11 |            |          |
|*  2 |   TABLE ACCESS FULL| SKEW |    32M|   335M| 31719   (37)| 00:00:43 |
-----------------------------------------------------------------------------

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

   2 - filter("COL1">0)
```

A Few Words on Hints!

They are finicky!
They are not well documented!
If you get it wrong, they are silently ignored! (grrrrrr!)
Aliases are important!
Query Block Names are important!
Complex Index Hints can be confusing!

```
INDEX(A SKEW_COL4_COL3)
INDEX_RS_ASC(@"SEL$1" "A"@"SEL$1" ("SKEW"."COL4” “SKEW”.”COL3”))
```

!!

Quick Digression on “Complex” Index Hints

Old Index Hint Format

INDEX(SKEW SKEW_IDX1)

New “Complex” Index Hint Format

INDEX_RS_ASC(@"SEL$1" "A"@"SEL$1" ("SKEW"."COL4” “SKEW”.”COL3”))

The hint is resolved as follows:

If an index name is specified, then the database only considered the specified index.

If a column list is specified, and if an index exists whose columns match the specified columns in number and order, then the database only consider this index. If no such index exists, then any index on the table with the specified columns as the prefix in the order specified is considered. In either case, the behavior is exactly as if the user had specified the same hint individually on all the matching indexes.

11.2.0.1 Performance Tuning Guide (chapter 19 – Using Optimizer Hints)

enkitec

A Few Words on Query Block Names

They are finicky!
They are not well documented!
If you get it wrong, they are silently ignored (grrrrrr!)
…

Default QB Names look like SEL$1, DEL$1, UPD$1, SEL$2 ...
Can be named using the qb_name hint (seldom used)
Probably best to look at existing hints (v$sql_plan.other_x

```
INDEX_RS_ASC(@"SEL$1" "A"@"SEL$1" ("SKEW"."COL4" "SKEW"."COL3"))
```

Translation

```
Index_Hint (Query_Block Table (Colum, …)
```

You can observe a lot by watching. ~ Yogi Bera

Hints are stored for every statement: OTHER_XML

```
SYS@LAB112> @other_xml
SYS@LAB112> select other_xml from v$sql_plan
  2  where sql_id like nvl('&sql_id', sql_id)
  3  and child_number like nvl('&child_number',child_number)
  4  and other_xml is not null
  5  /
Enter value for sql_id: 2gs7q8n2y7j76
Enter value for child_number: 0

OTHER_XML
--------------------------------------------------------------------------------
<other_xml><info type="db_version">11.2.0.1</info><info type="parse_schema"><![C
DATA["SYS"]]></info><info type="plan_hash">1946853647</info><info type="plan_has
h_2">28316188</info><peeked_binds><bind nam=":N2" pos="1" dty="1" csi="178" frm=
"1" mxl="32">4e</bind></peeked_binds><outline_data><hint><![CDATA[IGNORE_OPTIM_E
MBEDDED_HINTS]]></hint><hint><![CDATA[OPTIMIZER_FEATURES_ENABLE('11.2.0.1')]]></
hint><hint><![CDATA[DB_VERSION('11.2.0.1')]]></hint><hint><![CDATA[ALL_ROWS]]></
hint><hint><![CDATA[OUTLINE_LEAF(@"SEL$1")]]></hint><hint><![CDATA[INDEX_RS_ASC(
@"SEL$1" "SKEW"@"SEL$1" ("SKEW"."COL4"))]]></hint></outline_data></other_xml>

1 row selected.
```

Easier on the Eyes: SQL_HINTS.SQL

```
SYS@LAB112> @sql_hints
SYS@LAB112> select
  2  extractvalue(value(d), '/hint') as outline_hints
  3  from
  4  xmltable('/*/outline_data/hint'
  5  passing (
  6  select
  7  xmltype(other_xml) as xmlval
  8  from
  9  v$sql_plan
 10  where
 11  sql_id like nvl('&sql_id',sql_id)
 12  and child_number = &child_no
 13  and other_xml is not null
 14  )
 15  ) d;
Enter value for sql_id: 84q0zxfzn5u6s
Enter value for child_no: 0

OUTLINE_HINTS
--------------------------------------------------------------------------------------------------------------------
IGNORE_OPTIM_EMBEDDED_HINTS
OPTIMIZER_FEATURES_ENABLE('11.2.0.1')
DB_VERSION('11.2.0.1')
ALL_ROWS
OUTLINE_LEAF(@"SEL$1")
FULL(@"SEL$1" "SKEW"@"SEL$1")

6 rows selected.
```

dbms_xplan – alias format option

```
SYS@LAB112> !cat dplan_alias.sql
set lines 150
select * from table(dbms_xplan.display_cursor('&sql_id','&child_no','alias'))
/

SYS@LAB112> @dplan_alias
Enter value for sql_id: 84q0zxfzn5u6s
Enter value for child_no:

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------------------------------------------
SQL_ID  84q0zxfzn5u6s, child number 1
-------------------------------------
select avg(pk_col) from kso.skew where col1 = 136133

Plan hash value: 568322376

-----------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |       |       | 28360 (100)|          |
|   1 |  SORT AGGREGATE    |      |     1 |    24 |            |          |
|*  2 |   TABLE ACCESS FULL| SKEW |    35 |   840 | 28360   (1)| 00:05:41 |
-----------------------------------------------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1
   2 - SEL$1 / SKEW@SEL$1

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

   2 - filter("COL1"=136133)
```

SQL Profile Secret Sauce

The Main Ah Ha:

- import_sql_profile can be used to manually create a SQL Profile with any hints

Closely related concept:

- <outline_data> from other_xml can be used as a source of hints

```
dbms_sqltune.import_sql_profile(sql_text => cl_sql_text,
                                profile => ar_profile_hints,
                                category => '&category',
                                name => '&profile_name',
                                force_match => &force_matching,
                                replace => true);
```

enkitec

*SQL Profile Scripts (*trivial)

sql_profiles – lists profiles (dba_sql_profiles)
sql_profile_hints – lists hints associated with a profile
find_sql_using_profile - (v$sql where sql_profile is not null)
drop_sql_profile - (dbms_sql_tune.drop_sql_profile)
disable_sql_profile - (dbms_sql_tune.alter_sql_profile)
enable_sql_profile - (dbms_sql_tune.alter_sql_profile)
alter_sql_profile - (name, category, status, description, ~~fixed~~)

DEMO

SQL Profile Scripts (non-trivial)

create_sql_profile – uses OTHER_XML to create profile
create_sql_profile_awr – creates profile for plan in AWR history
move_sql_profile – copies a profile to another statement
create_1_hint_sql_profile – creates single line profile
gps.sql – creates a profile with the gather_plan_statistics hint

DEMO

STA Scripts

create_STA_task – creates a single statement task
accept_STA_profile – creates recommended profile
lock_STA_profile – removes opt_estimate hints

DEMO

enkitec

The Wrong Tool for the Job?

Maybe:

Certainly I'm proposing using Profiles in a way that was not originally intended.

import_sql_profile is not documented and could change (in version 12?).

It's easy to convert to Baselines.

I think benefits far outweigh the risks.

References

Tom Kyte. Pretty much everything he has ever written, but specifically
http://asktom.oracle.com/pls/asktom/f?p=100:11:0::::P11_QUESTION_ID:61313086268493

Jonathan Lewis. *Several Posts on Profiles*
http://jonathanlewis.wordpress.com/?s=%22sql+profile%22

Jonathan Lewis. *Plan Stability in Oracle 8i/9i*
http://www.dbazine.com/oracle/or-articles/jlewis4

Kerry Osborne. *Several Posts on Profiles*
http://kerryosborne.oracle-guy.com/

Randolf Geist. *Using Existing Cursors to Create Stored Outlines and SQL Profiles*
https://www.blogger.com/comment.g?blogID=5124641802818980374&postID=1108887738796239333

Notes on Editing Outlines on My Oracle Support (726802.1, 726802.1, 144194.1)
https:support.oracle.com

Questions / Contact Information

Questions?

Contact Information : Kerry Osborne

kerry.osborne@enkitec.com
kerryosborne.oracle-guy.com
www.enkitec.com