



**SAGE Computing Services**

Customised Oracle Training Workshops and Consulting

# the Cost Based Optimiser in 11g Rel 2

**Penny Cookson**

SAGE Computing Services

[www.sagecomputing.com.au](http://www.sagecomputing.com.au)

[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)



# **SAGE Computing Services**

Customised Oracle Training Workshops and Consulting

***Penny Cookson***

***Managing Director and Principal Consultant***

***Working with Oracle products since 1987***

***Oracle Magazine Educator of the Year 2004***



*[www.sagecomputing.com.au](http://www.sagecomputing.com.au)*

*[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)*



# Agenda

Extended Statistics

Kernel trace granularity - oradebug

Cardinality Feedback

Adaptive Cursors

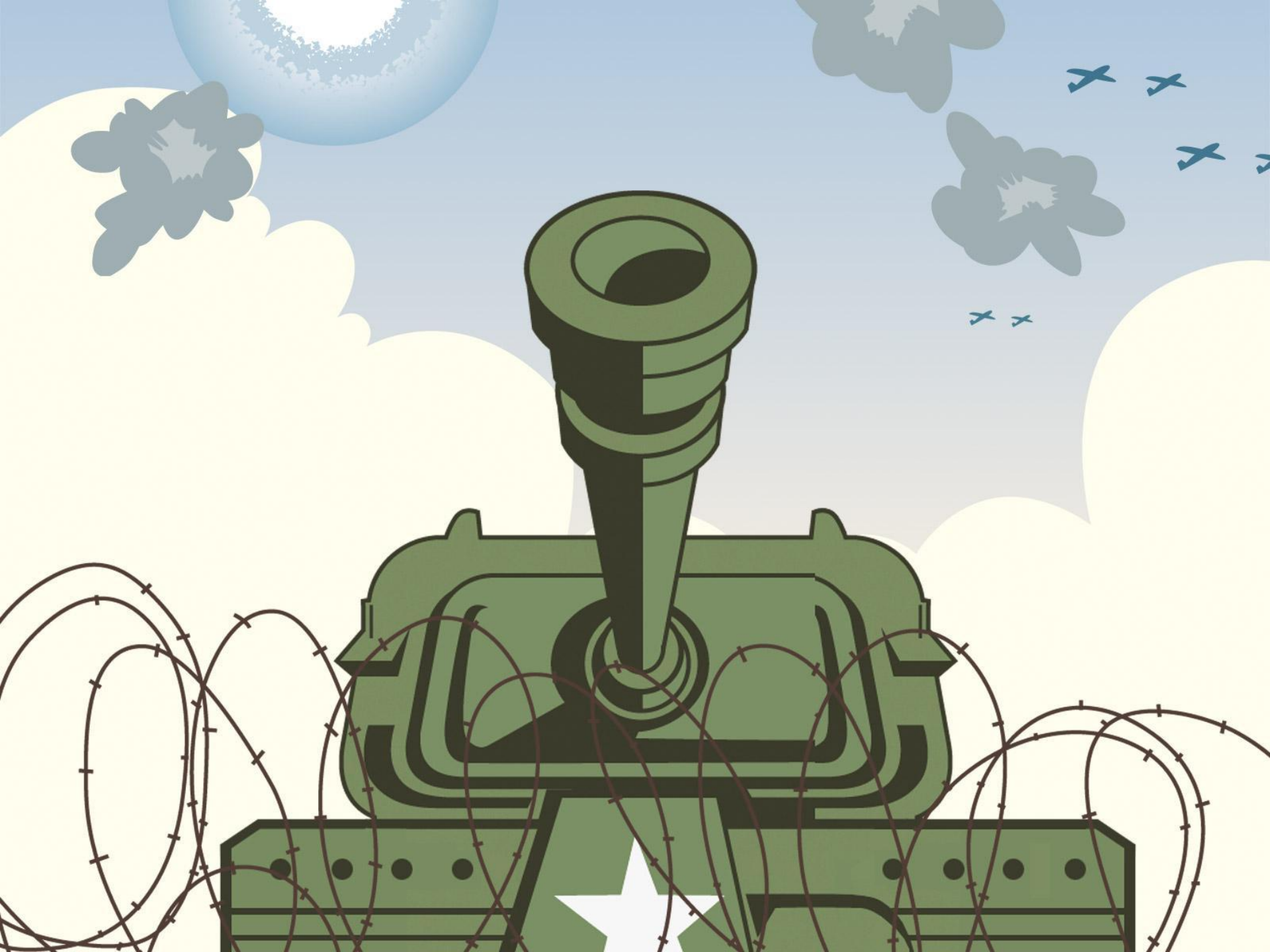
New behaviours

Hints in 11g

Getting rid of your hints

Assessing plan changes on upgrade

Results cache





## ***Gathering Statistics***

## Multi Column Statistics

Gather statistics on combinations of columns

Improves plans for complex dependency patterns  
(and this is persistent)

```
SELECT count(b.comments)
FROM train.events_large e, train.bookings_large b
WHERE e.org_id = 2264
AND   e.event_no = b.event_no
AND   e.comments = 'TEST'
```

# Multi Column Data Patterns

```
SELECT count(*)  
FROM train.events_large e
```

A Z	COUNT(*)
	100322

```
SELECT count(*)  
FROM train.events_large e  
WHERE e.org_id = 2264
```

A Z	COUNT(*)
	25083

```
SELECT count(e.comments)  
FROM train.events_large e  
WHERE e.comments = 'TEST'
```

A Z	COUNT(E.COMMENTS)
	75243

```
SELECT count(e.comments)  
FROM train.events_large e  
WHERE e.comments = 'TEST'  
AND e.org_id = 2264
```

A Z	COUNT(E.COMMENTS)
	4

# With Individual Statistics

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	2	5.00	22.52	35341	36337	0	1
total	4	5.00	22.52	35341	36337	0	1

Misses in library cache during parse: 1

Optimizer mode: ALL\_ROWS

Parsing user id: 82

Rows	Row Source Operation
1	SORT AGGREGATE (cr=36337 pr=35341 pw=35341 time=0 us)
175	HASH JOIN (cr=36337 pr=35341 pw=35341 time=13 us cost=9966 size=33365083 card=1076293)
4	<b>TABLE ACCESS FULL EVENTS_LARGE</b> (cr=991 pr=0 pw=0 time=3 us cost=274 size=319770 card=18810)
5767168	<b>TABLE ACCESS FULL BOOKINGS_LARGE</b> (cr=35346 pr=35341 pw=35341 time=225583 us cost=9665 size=80740352 card=5767168)

Elapsed times include waiting on following events:

Event waited on	Times Waited	Max. Wait	Total Waited
SQL*Net message to client	2	0.00	0.00
direct path read	314	0.38	15.88
resmgr:cpu quantum	11	0.14	0.27
SQL*Net message from client	2	0.04	0.04

\*\*\*\*\*






# Auto Created Column Statistics

```
BEGIN
```

```
  dbms_stats.gather_table_stats(ownname=>'TRAI1N',  
    tabname=>'EVENTS_LARGE', cascade=>TRUE,  
    method_opt => 'FOR ALL COLUMNS SIZE AUTO',  
    no_invalidate=>FALSE);
```

```
END;
```

```
SELECT column_name, histogram, num_buckets  
FROM   user_tab_col_statistics  
WHERE  table_name = 'EVENTS_LARGE'
```

 COLUMN_NAME	 HISTOGRAM	 NUM_BUCKETS
EVENT_NO	NONE	1
ORG_ID	FREQUENCY	6
DESCRIPTION	NONE	1
CONTACT_NAME	NONE	1
START_DATE	NONE	1
END_DATE	NONE	1
COMMENTS	FREQUENCY	2

# Auto Created Column Statistics

```
SELECT extension_name, extension
FROM   user_stat_extensions
WHERE  table_name='EVENTS_LARGE'
```

EXTENSION_NAME	EXTENSION

So same access path used

\*\*\*\*\*

Rows	Row	Source	Operation
-----	-----	-----	-----
	1	SORT	AGGREGATE (cr=36337 pr=35341 pw=35341 time=0 us)
	175	HASH	JOIN (cr=36337 pr=35341 pw=35341 time=13 us cost=9966 size=33365083 card=1076293)
	4	TABLE	ACCESS FULL EVENTS_LARGE (cr=991 pr=0 pw=0 time=3 us cost=274 size=319770 card=18810)
5767168		TABLE	ACCESS FULL BOOKINGS_LARGE (cr=35346 pr=35341 pw=35341 time=225583 us cost=9665 size=80740352 card=5767168)

# Manually Create Multi Column Statistics

BEGIN

```
dbms_stats.gather_table_stats(ownname=>'TRAIN',  
    tabname=>'EVENTS_LARGE', cascade=>TRUE,  
    method_opt => 'FOR COLUMNS (org_id, comments) SIZE  
    SKEWONLY', no_invalidate=>FALSE);
```

END;


OR

```
dbms_stats.create_extended_stats('TRAIN1','EVENTS_LARGE','(OR  
G_ID,COMMENTS)'); --(Then gather stats)
```

SELECT extension\_name, extension




FROM user\_stat\_extensions

WHERE table\_name='EVENTS\_LARGE'

 EXTENSION_NAME	EXTENSION
SYS_STUXRIN\$UJ2J7FKD#I4PEGV6BE	("ORG_ID","COMMENTS")

# Manually Create Multi Column Statistics

```
SELECT column_name, histogram, num_buckets  
FROM   user_tab_col_statistics  
WHERE  table_name = 'EVENTS_LARGE'
```

 COLUMN_NAME	 HISTOGRAM	 NUM_BUCKETS
EVENT_NO	NONE	1
ORG_ID	FREQUENCY	6
DESCRIPTION	NONE	1
CONTACT_NAME	NONE	1
START_DATE	NONE	1
END_DATE	NONE	1
COMMENTS	FREQUENCY	2
SYS_STUXRIN\$UJ...	FREQUENCY	8

# Multi Column Statistics

\*\*\*\*\*

```
SELECT count(b.comments)
FROM train.events_large e, train.bookings_large b
WHERE e.org_id = 2264
AND    e.event_no = b.event_no
AND    e.comments = 'TEST'
```

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	2	0.01	0.01	0	1175	0	1
total	4	0.01	0.01	0	1175	0	1

Rows	Row Source Operation
1	SORT AGGREGATE (cr=1175 pr=0 pw=0 time=0 us)
175	NESTED LOOPS (cr=1175 pr=0 pw=0 time=71 us)
175	NESTED LOOPS (cr=1001 pr=0 pw=0 time=39 us cost=1129 size=26815 card=865)
4	TABLE ACCESS FULL EVENTS_LARGE (cr=991 pr=0 pw=0 time=4 us cost=274 size=255 card=15)
175	<b>INDEX RANGE SCAN BK_EVT2</b> (cr=10 pr=0 pw=0 time=10 us cost=2 size=0 card=57)(object id 69868)
175	TABLE ACCESS BY INDEX ROWID BOOKINGS_LARGE (cr=174 pr=0 pw=0 time=0 us cost=57 size=798 card=57)

Elapsed times include waiting on following events:

Event waited on	Times Waited	Max. Wait	Total Waited
SQL*Net message to client	2	0.00	0.00
SQL*Net message from client	2	0.02	0.02

\*\*\*\*\*

# Multi Column Statistics

\*\*\*\*\*

```
SELECT count(b.comments)
FROM train.organisations o, train.events_large e, train.bookings_large b,
     train.resources r
WHERE o.org_id = e.org_id
AND   e.event_no = b.event_no
AND   b.resource_code = r.code
AND   o.name = 'Australian Medical Systems'
AND   r.description = 'Buffet Lunch'
AND   e.comments = 'TEST'
```

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	2	4.64	21.66	35341	36351	0	1
total	4	4.64	21.66	35341	36351	0	1

Misses in library cache during parse: 0  
Optimizer mode: ALL\_ROWS  
Parsing user id: 82

# Multi Column Statistics

\*\*\*\*\*

Rows      Row Source Operation

-----

```

      1  SORT AGGREGATE (cr=36351 pr=35341 pw=35341 time=0 us)
    25   HASH JOIN   (cr=36351 pr=35341 pw=35341 time=1966789 us cost=9972 size=6555490
card=79945)
      4   HASH JOIN   (cr=1005 pr=0 pw=0 time=8 us cost=281 size=792225 card=12575)
      1    MERGE JOIN CARTESIAN (cr=14 pr=0 pw=0 time=0 us cost=6 size=46 card=1)
      1     TABLE ACCESS FULL ORGANISATIONS (cr=7 pr=0 pw=0 time=0 us cost=3 size=26
card=1)
      1      BUFFER SORT (cr=7 pr=0 pw=0 time=0 us cost=3 size=20 card=1)
      1      TABLE ACCESS FULL RESOURCES (cr=7 pr=0 pw=0 time=0 us cost=3 size=20
card=1)
75243   TABLE ACCESS FULL EVENTS_LARGE (cr=991 pr=0 pw=0 time=1020 us cost=274
size=1282616 card=75448)
5767168  TABLE ACCESS FULL BOOKINGS_LARGE (cr=35346 pr=35341 pw=35341 time=350849 us
cost=9665 size=109576192 card=5767168)
```

Elapsed times include waiting on following events:

Event waited on	Times Waited	Max. Wait	Total Waited
-----	-----	-----	-----
SQL*Net message to client	2	0.00	0.00
direct path read	314	0.35	15.49
resmgr:cpu quantum	2	0.12	0.12
SQL*Net message from client	2	0.04	0.05

\*\*\*\*\*

# Multi Column Statistics - Summary

Manually create column groups and statistics on them

Complex joins may still need SQL Profiles

Use for dependent column such as Country, State

Use DBMS\_STATS.SEED\_COL\_USAGE to gather column  
usage stats

Use DBMS\_STATS.REPORT\_COL\_USAGE to report on col  
usage stats





## ***Gathering Statistics - Preferences***

# Setting Statistics Gathering Defaults

Obsolete:

GET\_PARAM

SET\_PARAM Procedure

Use:

GET\_PREFS Function

SET\_TABLE\_PREFS (set for individual tables)

SET\_GLOBAL\_PREFS – for new objects

# Table Prefs

PROCEDURE SET\_TABLE\_PREFS

Argument Name	Type	In/Out	Default?
-----	-----	-----	-----
OWNNAME	VARCHAR2	IN	
TABNAME	VARCHAR2	IN	
PNAME	VARCHAR2	IN	
PVALUE	VARCHAR2	IN	

## PNAME

CASCADE

DEGREE

ESTIMATE\_PERCENT

**METHOD\_OPT**

NO\_INVALIDATE

GRANULARITY

**PUBLISH**

**INCREMENTAL** (only scan partitions that have been changed)

**STALE\_PERCENT**

Database Instance: ora11.sagecomputing.com.au &gt; Tables &gt; Manage Optimizer Statistics &gt;

Logged in As SYS

## Global Statistics Gathering Options

Database ora11.sagecomputing.com.au

Cancel

Show SQL

Apply

## Statistics History

Retention Period  
(days)

31

The number of days for which optimizer statistics history will be retained.

## Gather Optimizer Statistics Default Options

Oracle recommends that you use the Gather Auto choice for the Gather Objects options when you use the Gather Optimizer Statistics process for Database and Schemas. If you choose not to use Gather Auto, the defaults for the other options are set here. Changing the options will impact the automated Optimizer Statistics Gathering task and user defined jobs.

Reset Defaults

Estimate Percentage



Auto (Oracle recommended)



100%



Percentage

Degree of Parallelism



Table default



Auto



System default



Degree

Granularity

Auto



Cursor Invalidation



Auto (Oracle recommended)



Immediate



None

Cascade



Auto (Oracle recommended)



True



False

Target Object Class (Auto Job)



Auto (Oracle recommended)



All



Oracle

Stale Percentage

10

Incremental



True



False

Publish



True



False

Histograms

FOR ALL COLUMNS SIZE AUTO



# Setting Statistics Gathering Defaults


**ORACLE Enterprise Manager 11g** Database Control [Setup](#) [Preferences](#) [Help](#) [Logout](#)

Database Instance: ora11 > [Object Level Statistics Gathering Preferences](#) > [Database](#) Logged in As SYS

## Add Table Preferences

[Show SQL](#) [Cancel](#) [OK](#)

**General** **Statistics Extension**

\* Table Name  

Estimate Percentage ☒ Inherit Global ☐ Auto (Oracle recommended) ☐ 100% ☐ Percentage

Degree of Parallelism ☒ Inherit Global ☐ Table default ☐ Auto ☐ System default ☐ Degree

Granularity

Cursor Invalidation ☒ Inherit Global ☐ Auto (Oracle recommended) ☐ Immediate ☐ None

Cascade ☒ Inherit Global ☐ Auto (Oracle recommended) ☐ True ☐ False

Stale Percentage ☒ Inherit Global ☐ Percentage

Incremental ☒ Inherit Global ☐ True ☐ False

Publish ☒ Inherit Global ☐ True ☐ False

Histograms ☒ Inherit Global ☐ FOR COLUMNS  
RESOURCE\_CODE SIZE 11

# Setting Statistics Gathering Defaults

**ORACLE** Enterprise Manager 11g
 Database Control
 [Setup](#) [Preferences](#) [Help](#) [Logout](#)

Database

Database Instance: ora11 > [Object Level Statistics Gathering Preferences](#) >
 Logged in As SYS

## Add Table Preferences

Show SQL

Cancel

OK

General

Statistics Extension

The statistics collected for column groups and expressions are called "extended statistics". Statistics on Column groups are used by optimizer for accounting correlation between columns. For example, if query has predicates c1=1 and c2=1 and if there are statistics on (c1, c2), optimizer will use this statistics for estimating the combined selectivity of the predicates. The expression statistics are used by optimizer for estimating selectivity of predicates on those expressions.

Extension	Type	Delete
(ORG_ID,COMMENTS)	User Defined	
	User Defined	
	User Defined	
	User Defined	
	User Defined	

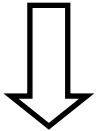
Add 5 Extensions

General

Statistics Extension

# Gathering and Publishing

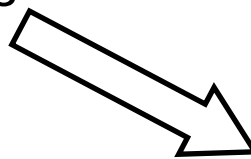
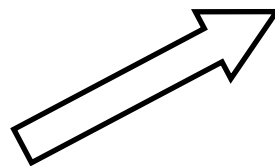
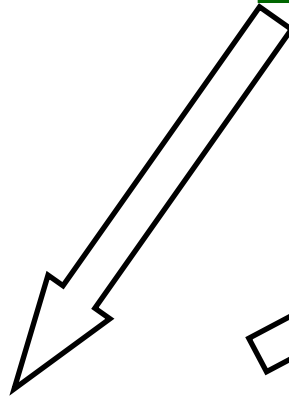
Set preferences to PUBLISH = 'FALSE'



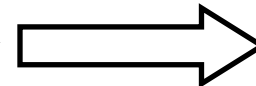
Gather Statistics

user\_tab\_pending\_stats  
user\_ind\_pending\_stats  
user\_col\_pending\_stats

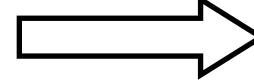
user\_tab\_statistics  
user\_ind\_statistics  
user\_tab\_col\_statistics



Better



Worse



dbms\_stats.  
publish\_pending\_stats

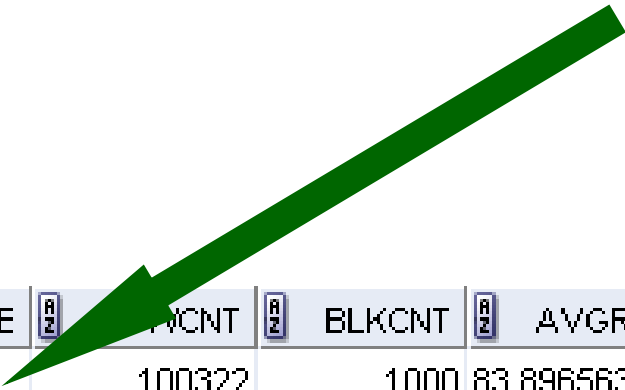
dbms\_stats.  
delete\_pending\_stats

Run test case  
Alter session set  
optimizer\_pending\_statistics  
= TRUE

Run test case

# Gathering and Publishing

```
SELECT obj#, TO_CHAR(savtime,'dd/mm/yyyy') save_time,  
       rowcnt, blkcnt, avgrln, samplesize, analyzetime  
FROM   wri$_optstat_tab_history  
ORDER BY savtime desc
```



OBJ#	SAVE_TIME	ROWCNT	BLKCNT	AVGRLN	SAMPLESIZE	ANALYZETIME
69846	01/12/2000	100322	1000	83.89656306...	100322	04/NOV/07
69670	04/11/2007	19344	244	46	19344	04/NOV/07
69656	04/11/2007	3833	118	230	3833	04/NOV/07
69742	04/11/2007	22122	95	26	22122	04/NOV/07



## Gathering and Publishing

Never publish any statistics until you  
are happy they work better

# Trace Granularity

Oracle 11g:

Improved Oracle diagnostic event infrastructure

[ORADEBUG DOC](#)





# Tracing SQL Executions

ALTER SESSION SET EVENTS

'sql\_trace[SQL: 324dr9k932njj]

plan\_stat=all\_executions,wait=true,bind=true';

SELECT count(b.comments)

FROM events\_large e, bookings\_large b

WHERE e.org\_id = 2264

AND e.event\_no = b.event\_no

AND e.comments = 'TEST'

Execute this twice and format the trace file

[ora11\\_ora\\_5852\\_324dr9k932njj.rtf.](#)



# Cardinality Feedback

**Real cardinality v expected cardinality?**

We have been able to find information about this with  
dbms\_xplan since 10g:

## Cardinality Feedback

```
SELECT /*+ GATHER_PLAN_STATISTICS */  
        count(b.comments)  
FROM events_large e, bookings_large b  
WHERE e.org_id = 2264  
AND   e.event_no = b.event_no  
AND   e.comments = 'TEST'
```

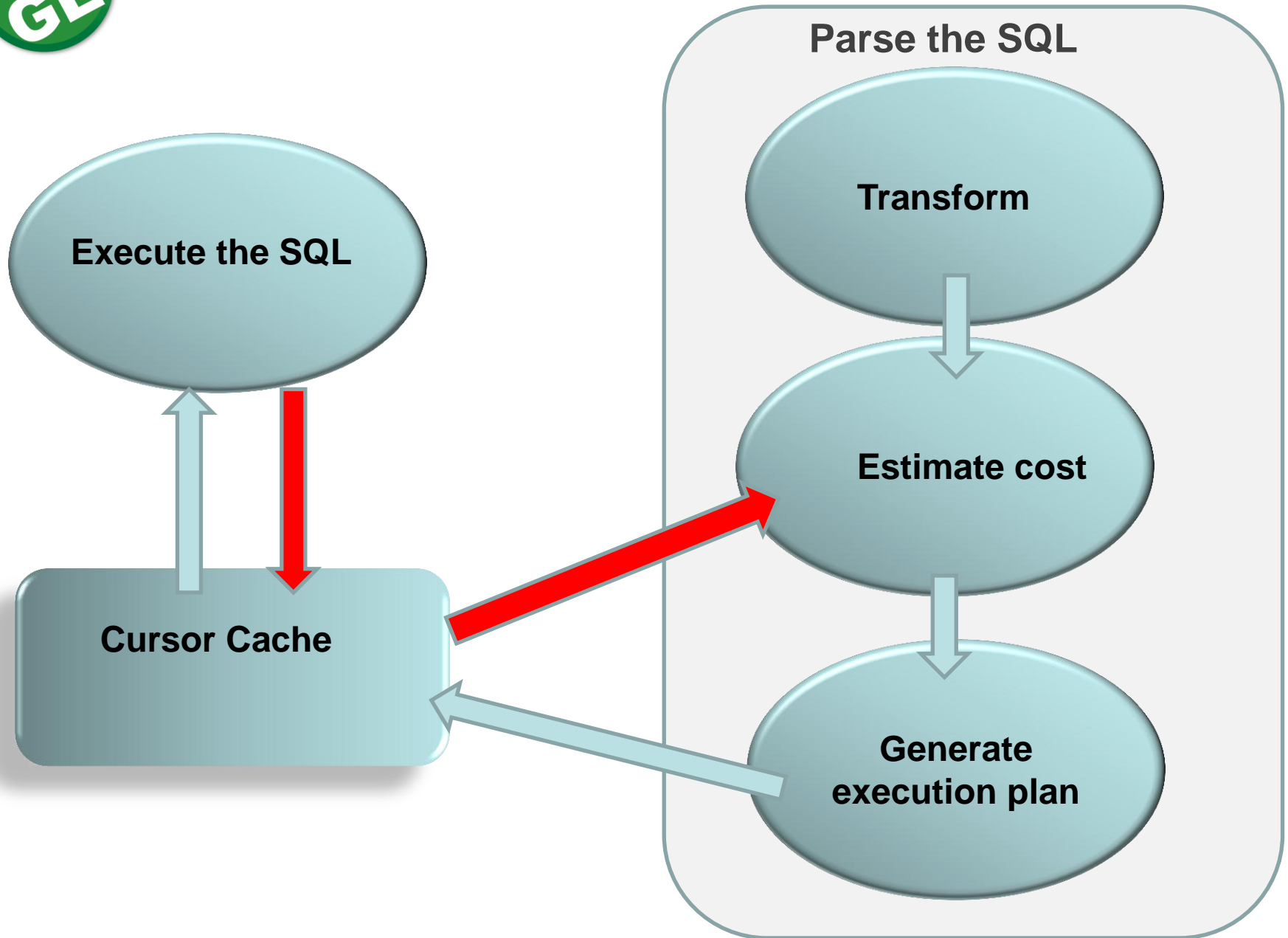
```
SELECT  
    DBMS_XPLAN.DISPLAY_CURSOR  
    (FORMAT=>'ALLSTATS LAST')  
FROM dual;
```



# Cardinality Feedback

```
SYS.DBMS_XPLAN_TYPE('SQL_ID 2yxvxc5r9z6tv, child number 0')
SYS.DBMS_XPLAN_TYPE('-----')
SYS.DBMS_XPLAN_TYPE('SELECT /*+ GATHER_PLAN_STATISTICS */ count(b.comments) FROM ')
SYS.DBMS_XPLAN_TYPE('events_large e, bookings_large b WHERE e.org_id = 2264 AND e.event_no ')
SYS.DBMS_XPLAN_TYPE('= b.event_no AND e.comments = 'TEST'')
SYS.DBMS_XPLAN_TYPE(' ')
SYS.DBMS_XPLAN_TYPE('Plan hash value: 3269294976')
SYS.DBMS_XPLAN_TYPE(' ')
SYS.DBMS_XPLAN_TYPE('-----')
-----')
SYS.DBMS_XPLAN_TYPE('| Id | Operation | Name | Starts | E-Rows | A-Rows | A-Time | Buffers | Reads |
OMem | lMem | Used-Mem |')
SYS.DBMS_XPLAN_TYPE('-----')
-----')
SYS.DBMS_XPLAN_TYPE('| 0 | SELECT STATEMENT | | 1 | | 1 | 00:00:02.23 | 36451 | 741 |
| | |')
SYS.DBMS_XPLAN_TYPE('| 1 | SORT AGGREGATE | | 1 | 1 | 1 | 00:00:02.23 | 36451 | 741 |
| | |')
SYS.DBMS_XPLAN_TYPE('|* 2 | HASH JOIN | | 1 | 398K | 232 | 00:00:13.89 | 36451 | 741 |
1452K | 1452K | 630K (0) |')
SYS.DBMS_XPLAN_TYPE('|* 3 | TABLE ACCESS FULL | EVENTS_LARGE | 1 | 6967 | 4 | 00:00:00.01 | 992 | 0 |
| | |')
SYS.DBMS_XPLAN_TYPE('| 4 | TABLE ACCESS FULL | BOOKINGS_LARGE | 1 | 5767K | 5767K | 00:00:01.64 | 35459 | 741 |
| | |')
SYS.DBMS_XPLAN_TYPE('-----')
-----')
SYS.DBMS_XPLAN_TYPE(' ')
SYS.DBMS_XPLAN_TYPE('Predicate Information (identified by operation id):')
SYS.DBMS_XPLAN_TYPE('-----')
SYS.DBMS_XPLAN_TYPE(' ')
SYS.DBMS_XPLAN_TYPE(' 2 - access("E"."EVENT_NO"="B"."EVENT_NO")')
SYS.DBMS_XPLAN_TYPE(' 3 - filter(("E"."ORG_ID"=2264 AND "E"."COMMENTS"='TEST'))')
SYS.DBMS_XPLAN_TYPE(' ')
```

# Cardinality Feedback





# Cardinality Feedback

If it has to do this it has got it wrong once

and

Its not persistent so it will keep getting it wrong once



# Cardinality Feedback

**Create your statistics so Oracle does not need to do this**

```
SELECT count(b.comments)
FROM events_large e, bookings_large b
WHERE e.org_id = 2264
AND   e.event_no = b.event_no
AND   e.comments = 'TEST'
```

```
dbms_stats.create_extended_stats
('TRAIN1','EVENTS_LARGE','(ORG_ID,COMMENTS)');
```

## Cardinality Feedback

**Set optimizer\_dynamic\_sampling so Oracle does not need to do this**

```
SELECT /*+ dynamic_sampling (4) */ count(b.comments)
FROM events_large e, bookings_large b
WHERE e.org_id = 2264
AND   e.event_no = b.event_no
AND   e.comments = 'TEST'
```

# Cardinality Feedback

## No use across multiple tables

```
SELECT count(b.comments)
FROM organisations o, events_large e,
      bookings_large b
WHERE o.org_id = e.org_id
AND   e.event_no = b.event_no
AND   o.name = 'Australian Medical Systems'
AND   e.comments = 'TEST'
```

# Cardinality Feedback

**Use optimizer\_scaling to correct multi table estimates**

```
SELECT /*+ opt_estimate(JOIN,("E","O"),  
          SCALE_ROWS= 0.000001) */ count(b.comments)  
FROM organisations o, events_large e,  
     bookings_large b  
WHERE o.org_id = e.org_id  
AND   e.event_no = b.event_no  
AND   o.name = 'Australian Medical Systems'  
AND   e.comments = 'TEST'
```

# Cardinality Feedback

Was it used?

The screenshot shows the Oracle SQL Developer interface. At the top, there are tabs for 'train1' and 'ora11'. Below the tabs is a toolbar with various icons. The main area displays a SQL query:

```

35
36 select sql_id,child_number, use_feedback_stats
37 from v$sql_shared_cursor
38 WHERE sql_id = '324dr9k932njj'
39

```

Below the query editor, there are two tabs: 'Statement Output' and 'Query Result'. The 'Query Result' tab is active, showing the results of the query. The status bar indicates 'All Rows Fetched: 2 in 0.007 seconds'.

	SQL_ID	CHILD_NUMBER	USE_FEEDBACK_STATS
1	324dr9k932njj	0	Y
2	324dr9k932njj	1	N

# Cardinality Feedback

## Oracle asking Did I get it right?

Need stable data patterns

Can use where poor stats and no dynamic sampling

Designed to correct mistakes

Not for bind variables

Only once per plan

Not persistent

# A Characteristic Problem

I haven't changed anything

Its really slow this morning

I did the same thing yesterday and it was fine

Actually its OK now

No its not

Thank you so much you've fixed it (I haven't done anything)



## Bind Peeking + Adaptive Cursors

```
SELECT resource_code, count(*)
FROM   bookings_large
GROUP BY resource_code;
```

RESOURCE_CODE	COUNT(*)
VCR1	497009
CONF	497019
LNCH	745520
BRSM	745524
PC1	48510
FLPC	497015
BRLG	1991051
TAP1	248509
VCR2	497011



Minority value



Majority value



## Bind Peeking

Look at the value of the bind variable when the statement is parsed

Plan is based on that value

```
BEGIN :v3 := 'PC1'; END;
```

```
SELECT COUNT(quantity)  
FROM bookings_large  
WHERE resource_code = :v3;
```



**INDEX  
for <1% of rows**

```
BEGIN :v3 := 'BRLG'; END;
```

```
SELECT COUNT(quantity)  
FROM bookings_large  
WHERE resource_code = :v3;
```



**SAME PLAN  
→ STILL INDEX  
for 34% of rows**

# Bind Peeking + Adaptive Cursors

```
SELECT COUNT(l.quantity) FROM train.bookings_large l  
WHERE resource_code = :v1
```

```
SELECT sql_id FROM v$sqlarea  
WHERE sql_text = 'SELECT COUNT(l.quantity)  
FROM train.bookings_large l WHERE resource_code = :v1'
```

SQL_ID
95jktg3mza0qm

# Bind Peeking + Adaptive Cursors

```
BEGIN :v1 := 'PC1'; END;
SELECT COUNT(l.quantity) FROM train.bookings_large l
WHERE resource_code = :v1

SELECT sql_id, child_number, is_bind_sensitive, is_bind_aware
FROM v$sql
WHERE sql_id = '95jktg3mza0qm';
```

SQL_ID	CHILD_NUMBER	IS_BIND_SENSITIVE	IS_BIND_AWARE
95jktg3mza0qm	0	Y	N

**IS\_BIND\_SENSITIVE** - may need to change the plan  
 - set on first execution of the statement

**IS\_BIND\_AWARE** - do need to change the plan

# Bind Peeking + Adaptive Cursors

```
SELECT child_number, bind_set_hash_value, peeked, executions,
       rows_processed, buffer_gets
FROM   v$sql_cs_statistics
WHERE  sql_id ='95jktg3mza0qm'
```

CHILD_NUMBER	BIND_SET_HASH_VALUE	PEEKED	EXECUTIONS	ROWS_PROCESSED	BUFFER_GETS
0	1466228028	Y	1	291062	6746



# Bind Peeking + Adaptive Cursors

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.01	0.01	0	0	0	0
Fetch	2	0.12	0.43	119	6635	0	1
total	4	0.14	0.44	119	6635	0	1

Misses in library cache during parse: 1  
Misses in library cache during execute: 1  
Optimizer mode: ALL\_ROWS  
Parsing user id: 82

Rows	Row	Source	Operation
	1	SORT AGGREGATE (cr=6635 pr=119 pw=119 time=0 us)	
	48510	TABLE ACCESS BY INDEX ROWID BOOKINGS_LARGE (cr=6635 pr=119 pw=119 time=2870 us cost=1977 size=316008 card=39501)	
	48510	<b>INDEX RANGE SCAN BK_RES2</b> (cr=104 pr=0 pw=0 time=527 us cost=90 size=0 card=39501) (object id 69870)	

\*\*\*\*\*

Elapsed times include waiting on following events:

Event waited on	Times	Max. Wait	Total Wait
	Waited		
SQL*Net message to client	2	0.00	0.00
SQL*Net message from client	2	0.00	0.00

\*\*\*\*\*

# Bind Peeking + Adaptive Cursors

```
BEGIN :v1 := 'BRLG'; END;
SELECT COUNT(l.quantity) FROM train.bookings_large l
WHERE resource_code = :v1
```

```
SELECT sql_id, child_number, is_bind_sensitive, is_bind_aware
FROM v$sql
WHERE sql_id = '95jktg3mza0qm';
```

SQL_ID	CHILD_NUMBER	IS_BIND_SENSITIVE	IS_BIND_AWARE
95jktg3mza0qm	0	Y	N

# Bind Peeking + Adaptive Cursors

```
SELECT child_number, bind_set_hash_value, peeked, executions,
       rows_processed, buffer_gets
FROM   v$sql_cs_statistics
WHERE  sql_id ='95jktg3mza0qm'
```

CHILD_NUMBER	BIND_SET_HASH_VALUE	PEEKED	EXECUTIONS	ROWS_PROCESSED	BUFFER_GETS
0	1466228028	Y	1	291062	6746



# Bind Peeking + Adaptive Cursors

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	2	6.29	78.82	34335	39594	0	1
total	4	6.29	78.82	34335	39594	0	1

Misses in library cache during parse: 0  
Optimizer mode: ALL\_ROWS  
Parsing user id: 82

Rows	Row Source Operation
1	SORT AGGREGATE (cr=39594 pr=34335 pw=34335 time=0 us)
1991051	TABLE ACCESS BY INDEX ROWID BOOKINGS_LARGE (cr=39594 pr=34335 pw=34335 time=626800 us cost=1977 size=316008 card=39501)
1991051	<b>INDEX RANGE SCAN BK_RES2</b> (cr=4437 pr=4436 pw=4436 time=61616 us cost=90 size=0 card=39501)(object id 69870)

Elapsed times include waiting on following events:

Event waited on	Times	Max. Wait	Total Wait
	Waited		
SQL*Net message to client	2	0.00	0.00
db file sequential read	34335	0.29	72.08
resmgr:cpu quantum	30	0.10	1.29
SQL*Net message from client	2	0.00	0.00

\*\*\*\*\*



# Bind Peeking + Adaptive Cursors

```
BEGIN :v1 := 'BRLG'; END;
SELECT COUNT(l.quantity) FROM train.bookings_large l
WHERE resource_code = :v1
```

```
SELECT sql_id, child_number, is_bind_sensitive, is_bind_aware
FROM v$sql
WHERE sql_id = '95jktg3mza0qm';
```

SQL_ID	CHILD_NUMBER	IS_BIND_SENSITIVE	IS_BIND_AWARE
95jktg3mza0qm	0	Y	N
95jktg3mza0qm	1	Y	Y

# Bind Peeking + Adaptive Cursors

```
SELECT child_number, bind_set_hash_value, peeked, executions,
       rows_processed, buffer_gets
FROM   v$sql_cs_statistics
WHERE  sql_id = '95jktg3mza0qm'
```

CHILD_NUMBER	BIND_SET_HASH_VALUE	PEEKED	EXECUTIONS	ROWS_PROCESSED	BUFFER_GETS
1	2982103524	Y	1	3982104	35346
0	1466228028	Y	1	291062	6746



# Bind Peeking + Adaptive Cursors

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	2	1.23	21.71	35341	35346	0	1
total	4	1.23	21.72	35341	35346	0	1

Misses in library cache during parse: 0  
Misses in library cache during execute: 1  
Optimizer mode: ALL\_ROWS  
Parsing user id: 82

Rows	Row Source Operation
1	SORT AGGREGATE (cr=35346 pr=35341 pw=35341 time=0 us)
1991051	<b>TABLE ACCESS FULL BOOKINGS_LARGE</b> (cr=35346 pr=35341 pw=35341 time=414975 us cost=9652 size=15700664 card=1962583)

Elapsed times include waiting on following events:

Event waited on	Times	Max. Wait	Total Wait
	Waited		
SQL*Net message to client	2	0.00	0.00
direct path read	314	0.42	19.45
SQL*Net message from client	2	0.10	0.10

\*\*\*\*\*

# Bind Peeking + Adaptive Cursors in 11g

Statements with bind variables (+histograms) are bind sensitive

The first time you execute a statement with different selectivity it uses the original plan

The second time it changes the plan and become bind aware

New values will use a plan for the appropriate selectivity range

# Adaptive Cursors Views

## V\$SQL

SQL_ID	CHILD_NUMBER	IS_BIND_SENSITIVE	IS_BIND_AWARE
7bmrs67hubufy	0	Y	N
7bmrs67hubufy	1	Y	Y
7bmrs67hubufy	2	Y	Y
7bmrs67hubufy	3	Y	Y

## V\$SQL\_CS\_SELECTIVITY

SQL_ID	CHILD_NUMBER	SUBSTR(PREDICATE,1,10)	RANGE_ID	LOW	HIGH
7bmrs67hubufy	1	=V1	0	0.008765	0.010712
7bmrs67hubufy	2	=V1	0	0.004382	0.010712
7bmrs67hubufy	3	=V1	0	0.004382	0.096808

# Adaptive Cursors Views

## V\$SQL\_CS\_HISTOGRAM

SQL_ID	CHILD_NUMBER	BUCKET_ID	COUNT
7bmrs67hubufy	0	1	1
7bmrs67hubufy	0	0	0
7bmrs67hubufy	0	2	1
7bmrs67hubufy	1	1	1
7bmrs67hubufy	1	0	0
7bmrs67hubufy	1	2	0
7bmrs67hubufy	2	1	0
7bmrs67hubufy	2	0	1
7bmrs67hubufy	2	2	0
7bmrs67hubufy	3	1	1
7bmrs67hubufy	3	0	11
7bmrs67hubufy	3	2	1

## V\$SQL\_CS\_STATISTICS

SQL_ID	CHILD_NUMBER	BIND_SET_HASH_VALUE	PEEKED	EXECUTIONS	ROWS_PROCESSED	BUFFER_GETS
7bmrs67hubufy	0	2982103524	Y	1	1991053	35333
7bmrs67hubufy	1	1466228028	Y	1	145519	7058
7bmrs67hubufy	2	778730927	Y	1	1	3
7bmrs67hubufy	3	2421911073	N	1	299998	4784
7bmrs67hubufy	3	3254803217	Y	1	1490902	34471



# PLSQL has soft parse avoidance

## No default adaptive cursor functionality in a session

Implicit cursor

Explicit cursor

Native dynamic SQL

## Default adaptive cursor functionality in a session

Session\_cached\_cursors = 0

Ref cursors

**Once another session executes the statement it will adapt**

# Adaptive Cursors Functionality

## Adaptive Cursors 11.1.0.6

Bind variable with Equality and Histogram  
Not for range conditions

## Adaptive Cursors 11.1.0.7 and 11.2

Bind variable with Equality and Histogram  
Range conditions  
Does not support LIKE  
`/*+ BIND_AWARE */`

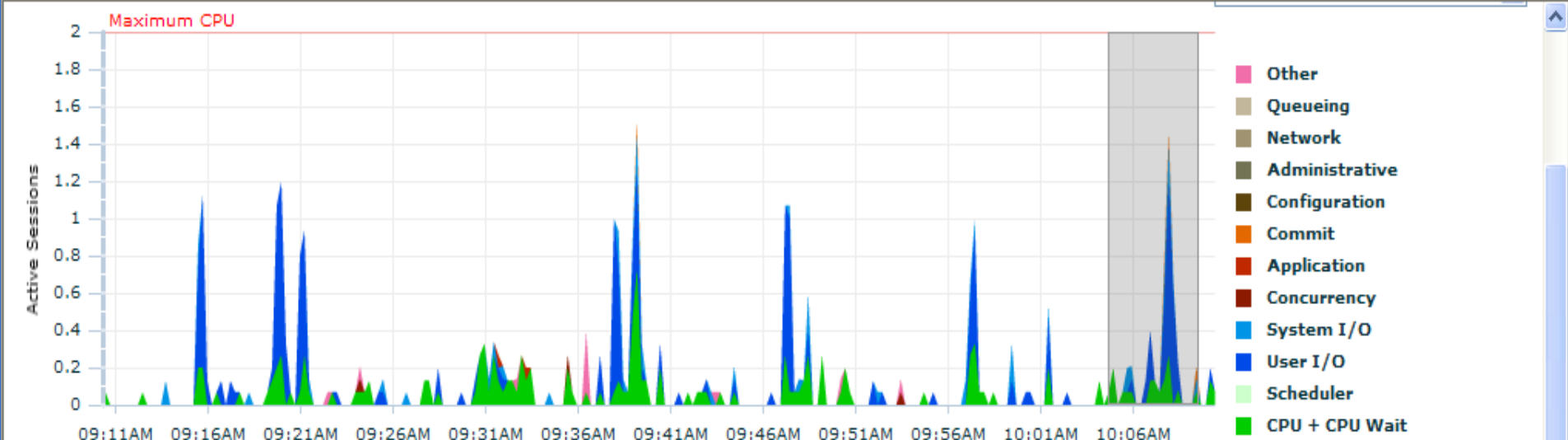




## So where are we now?

**Our own code – works properly first time with hint in SQL and PL/SQL**

**Software Packages – will still get it wrong once  
and won't use adaptive cursors for PL/SQL  
at all within one session  
unless we set session cached cursors =0  
or ref cursors**



## Detail for Selected 5 Minute Interval

Start Time 7/08/2010 10:04:41 AM GMT+08:00

Run ASH Report

### Top SQL

Actions

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
<input checked="" type="checkbox"/>	35.00	7bmrs67hubufy	SELECT
<input checked="" type="checkbox"/>	30.00	7bmrs67hubufy	SELECT
<input type="checkbox"/>	3.33	fndjrj10u6q7d	SELECT
<input type="checkbox"/>	1.67	6129566gyvx21	SELECT
<input type="checkbox"/>	1.67	cyzum8t19p208	SELECT
<input type="checkbox"/>	1.67	5955d0xup873q	SELECT
<input type="checkbox"/>	1.67	fwcwrxs43pz9	SELECT
<input type="checkbox"/>	1.67	5h0z49srcg7hr	SELECT
<input type="checkbox"/>	1.67	b6sgjk84vcnmp	SELECT

### Top Sessions

View

Activity (%)	Session ID	User Name	Program
46.99	70	TRAIN1	sqlplus.exe
9.64	6	SYS	ORACLE.EXE (CKPT)
7.23	2	SYS	ORACLE.EXE (GEN0)
6.02	68	SYSMAN	OMS
4.82	55	SYS	ORACLE.EXE (LGWR)
3.61	21	DBSNMP	OMS
3.61	20	SYSMAN	OMS
3.61	69	SYS	OMS
2.41	27	SYS	ORACLE.EXE (J000)
2.41	58	SYSMAN	OMS

Total Sample Count: 83

```

SELECT COUNT(l.quantity)
FROM train1.bookings_large l
WHERE resource_code = :v1
  
```

## Details

Select the plan hash value to see the details below.

Plan Hash Value

1457437069

There are multiple plans found for this SQL statement.

[Statistics](#)

**Activity**

[Plan](#)

[Plan Control](#)

[Tuning History](#)

All

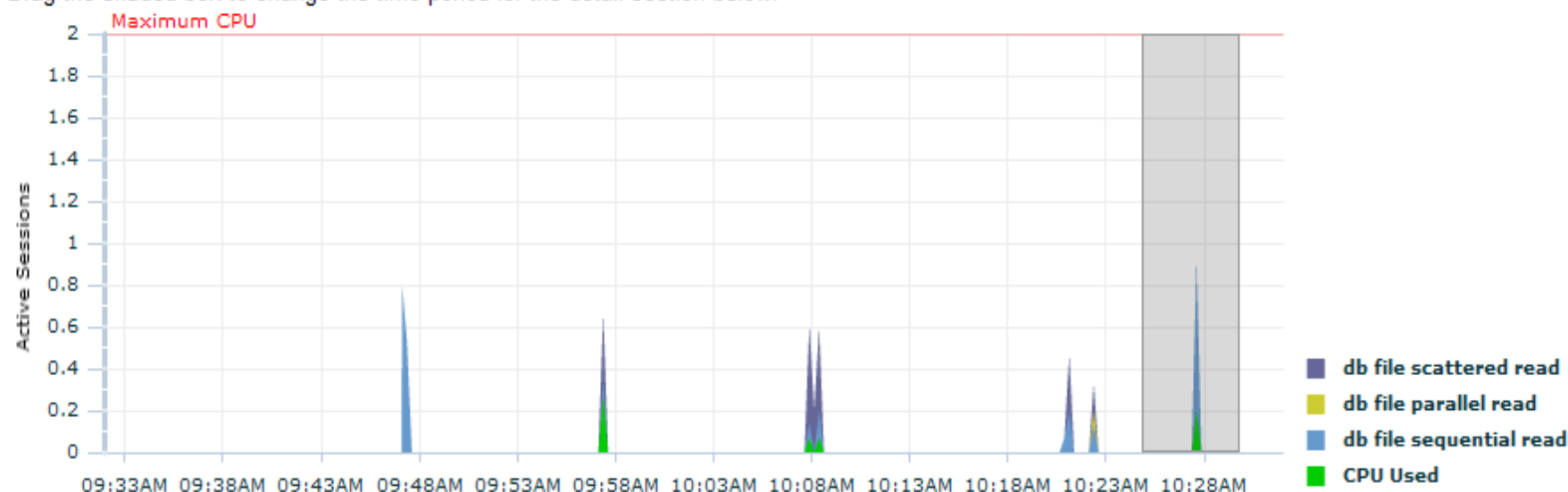
927983165

1457437069

[Tuning](#)

## Summary

Drag the shaded box to change the time period for the detail section below.



## Detail for Selected 5 Minute Interval

Start Time 7/08/2010 10:24:52

[Run AWR SQL Report](#)

[Run ASH Report](#)

Activity (%)	SID	QC SID	User	Program	Service	Plan Hash Value
100.00	19		TRAIN1	sqlplus.exe	ora11gr2.sagecomputing.com.au	1457437069

Navigation bar showing the URL: <https://sage7.sagecomputing.com.au:5505/em/console/database/instance/sqlDetail?ev>. A "Certificate Error" warning is present. The browser's address bar also displays "Live Search".

File Edit View Favorites Tools Help

Oracle Enterprise Manager (SYS) - SQL Details: 7bmrs...

Home RSS Print Page Tools

## Details

Select the plan hash value to see the details below. Plan Hash Value  There are multiple plans found for this SQL statement.

[Statistics](#) [Activity](#) **Plan** [Plan Control](#) [Tuning History](#) [SQL Monitoring](#)

Data Source **Cursor Cache** Capture Time **7/08/2010 10:14:39 (UTC+08:00)** Parsing Schema **TRAIN1** Optimizer Mode **ALL\_ROWS**

Additional Information

View ☒ Graph ☐ Table





## SQL Details: 7bmrs67hubufy

Switch to SQL ID

Go

View Data

Real Time: Manual Refresh



Refresh

SQL Worksheet

Schedule SQL Tuning Advisor

SQL Repair Ad

Text



```
SELECT COUNT(l.quantity)
FROM train1.bookings_large l
WHERE resource_code = :v1
```

## Details

Select the plan hash value to see the details below.

Plan Hash Value

927983165

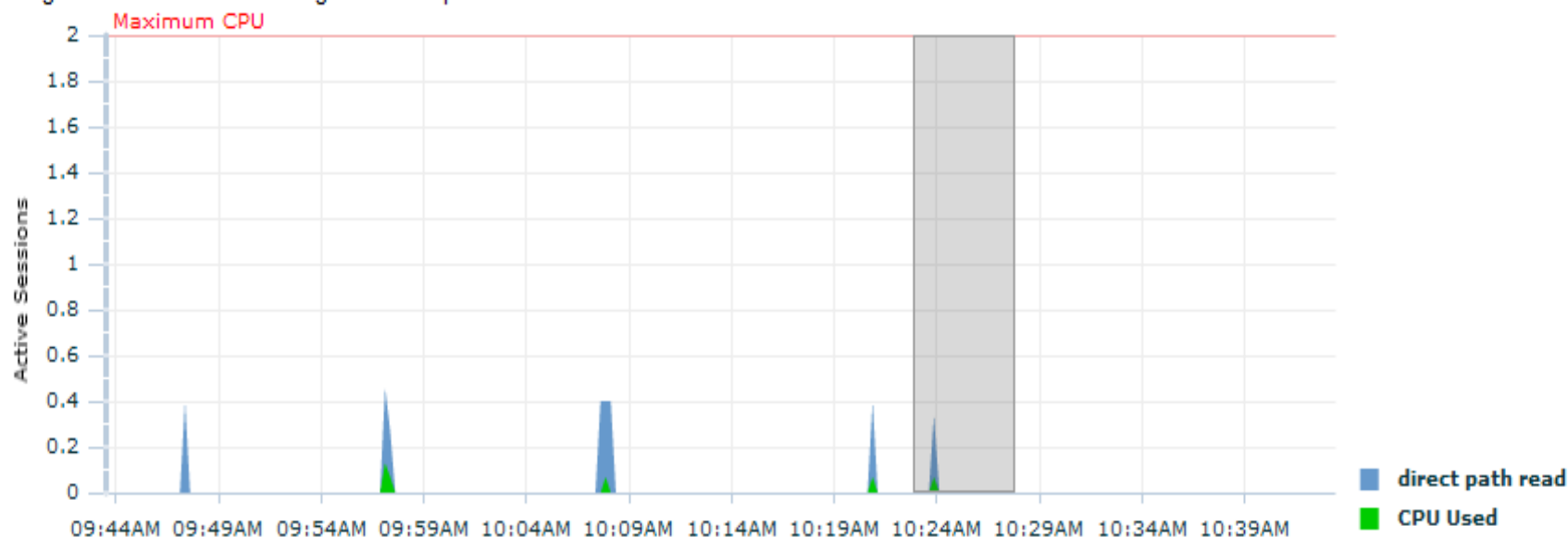


There are multiple plans found for this SQL statement.

[Statistics](#)**Activity**[Plan](#)[Plan Control](#)[Tuning History](#)[SQL Monitoring](#)

## Summary

Drag the shaded box to change the time period for the detail section below.



Database Instance: ora11gr2.sagecomputing.com.au > [Top Activity](#) >

Logged in As SYS

## SQL Details: 7bmrs67hubufy

Switch to SQL ID   View Data Real Time: Manual Refresh

[Text](#)

```
SELECT COUNT(l.quantity)
FROM train1.bookings_large l
WHERE resource_code = :v1
```

### Details

Select the plan hash value to see the details below. Plan Hash Value 927983165  There are multiple plans found for this SQL statement.

[Statistics](#) [Activity](#) **Plan** [Plan Control](#) [Tuning History](#) [SQL Monitoring](#)

Data Source **Cursor Cache** Capture Time 7/08/2010 10:33:52 (UTC+08:00) Parsing Schema **TRAIN1** Optimizer Mode **ALL\_ROWS**

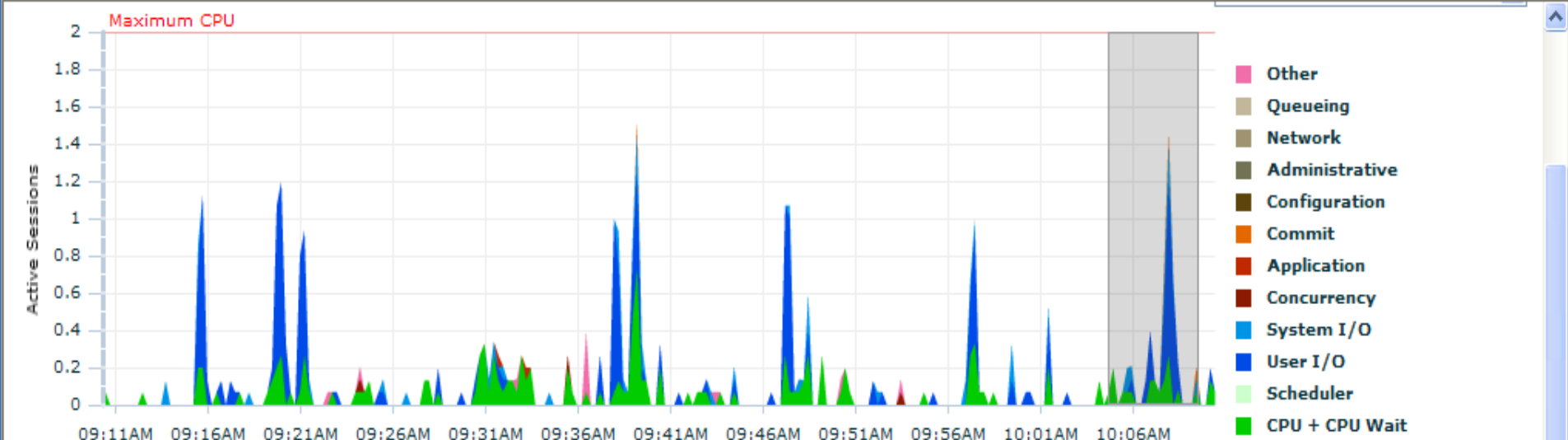
Additional Information

View ☒ Graph ☐ Table

Overview

Selection Details





### Detail for Selected 5 Minute Interval

Start Time 7/08/2010 10:04:41 AM GMT+08:00

Run ASH Report

#### Top SQL

Actions

Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
<input checked="" type="checkbox"/>	35.00	7bmrs67hubufy	SELECT
<input checked="" type="checkbox"/>	30.00	7bmrs67hubufy	SELECT
<input type="checkbox"/>	3.33	fndj10u6q7d	SELECT
<input type="checkbox"/>	1.67	6129566gyx21	SELECT
<input type="checkbox"/>	1.67	cyzum8t19p208	SELECT
<input type="checkbox"/>	1.67	5955d0xup873q	SELECT
<input type="checkbox"/>	1.67	fwcwrxs43pz9	SELECT
<input type="checkbox"/>	1.67	5h0z49srcg7hr	SELECT
<input type="checkbox"/>	1.67	b6sgjk84vcnmp	SELECT

#### Top Sessions

View

Activity (%)	Session ID	User Name	Program
46.99	70	TRAIN1	sqlplus.exe
9.64	6	SYS	ORACLE.EXE (CKPT)
7.23	2	SYS	ORACLE.EXE (GEN0)
6.02	68	SYSMAN	OMS
4.82	55	SYS	ORACLE.EXE (LGWR)
3.61	21	DBSNMP	OMS
3.61	20	SYSMAN	OMS
3.61	69	SYS	OMS
2.41	27	SYS	ORACLE.EXE (J000)
2.41	58	SYSMAN	OMS

Total Sample Count: 83



Cancel

Submit

Specify the following parameters to schedule a job to run the SQL Tuning Advisor.

\* Name

Description

\* SQL Tuning Set



SQL Tuning Set Description **Automatically generated by Top SQL**

SQL Statements Counts 2

### ► SQL Statements

### Scope

Total Time Limit  
(minutes)

Scope of Analysis ☐ Limited

The analysis is done without SQL Profile recommendation and takes about 1 second per statement.

☒ Comprehensive

This analysis includes SQL Profile recommendation, but may take a long time.

Time Limit per Statement (minutes)

### Schedule

Time Zone

☒ Immediately

☐ Later

Date   
(example: 7/08/2010)



Time    ☒ AM ☐ PM

Cancel

Submit



# Recommendations for SQL ID:7bmrs67hubufy

Return

Only one recommendation should be implemented.

## SQL Text

SELECT COUNT(l.quantity) FROM train1.bookings\_large l WHERE resource\_code = :v1

## Select Recommendation

Original Explain Plan (Annotated)

Select	Type	Findings	Recommendations	Rationale	Benefit (%)	Other Statistics	New Explain Plan	Compare Explain Plans
	Alternative Plans	Some alternative execution plans for this statement were found by searching the system's real-time and historical performance data.	The Original Plan appears to have the best performance, based on the elapsed time per execution. However, if you know that one alternative plan is better than the Original Plan, you can create a SQL plan baseline for it. This will instruct the Oracle optimizer to pick it over any other choices in the future.	Creating a plan baseline for the plan with the best elapsed time will prevent the Oracle optimizer from selecting a plan with worse performance.				

Return

Database Instance: ora11qr2.sagecomputing.com.au > [Advisor Central](#) > [SQL Tuning Summary:SQL\\_TUNING\\_1281148518078](#) >[SQL Tuning Details:SQL\\_TUNING\\_1281148518078](#) >

Logged in As SYS

## Recommendations for SQL ID:7bmrs67hubufy

[Return](#)

Only one recommendation should be implemented.

### SQL Text

[SELECT COUNT\(l.quantity\) FROM train1.bookings\\_large l WHERE resource\\_code = :v1](#)

### Select Recommendation

[Original Explain Plan \(Annotated\)](#)[Implement](#)**Maybe you should buy an Exadata box**

Select	Type	Findings	Recommendations	Rationale	Benefit (%)	Other Statistics	Explain Plan	Compare Explain Plans
<input checked="" type="radio"/>	Alternative Plans	Some alternative execution plans for this statement were found by searching the system's real-time and historical performance data.	Consider creating a SQL plan baseline for the plan with the best average elapsed time.	Execution statistics from the cursor cache show an average elapsed time of 4.971s for the recommended plan, compared to 6.86s for the original plan. Creating a plan baseline for the plan with the best elapsed time will prevent the Oracle optimizer from selecting a plan with worse performance.				

[Return](#)

**So Much Promise**



**And then the  
disappointment**





# Bind Peeking + Adaptive Cursors Summary

## Your code

Use the `BIND_AWARE` hint (SQL and PL/SQL) for skewed data

Use ref cursors (if hints are not allowed)

## Packages

PL/SQL – set `session_cached_cursors = 0` (temporarily)

Minimise statement invalidations

Consider running the statement with minority and majority value on start up



# Adaptive Cursors What we Really Need

## Adaptive Cursors Persistence

Once a statement has been bind aware it knows next time its parsed



# Try to get rid of your hints

## `_optimizer_ignore_hints`

Create additional statistics

Set `_optimizer_ignore_hints` to TRUE

Test the app

If it runs OK remove your hints

## 68 New Hints

```
SELECT name, inverse, sql_feature, version  
FROM v$sql_hint  
WHERE version like '11%'  
ORDER BY version desc, name
```

[New Hints](#)



# Much More Query Transformation

```
SELECT e.event_no, e.start_date, sum(cost) totcost
FROM   events_large e, bookings_large b
WHERE  e.event_no = b.event_no
GROUP BY e.event_no, e.start_date
```

alter session set tracefile\_identifier = Penny

```
ora11gr2_trace10053.trc - WordPad
File Edit View Insert Format Help

Query Block Registry:
SEL$1 0x0 (PARSER)
  SEL$54B5B641 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST LIST 3)
  SEL$54B5B641 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST LIST 3)
  SEL$706665FA 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST 2)
  SEL$706665FA 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST 2)
  SEL$38F5D95B 0x0 (QUERY BLOCK TABLES CHANGED SEL$1)
    SEL$137A03FC 0x0 (SPLIT/MERGE QUERY BLOCKS SEL$38F5D95B)
  SEL$7C398D44 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST 5) [FINAL]
    SEL$7A931295 0x0 (PUSHED PREDICATE SEL$7D2C682D; SEL$7C398D44; "VW_GBC_5"@"SEL$F486F43F" 1)
  SEL$7C398D44 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST 5) [FINAL]
  ...
  SEL$F486F43F 0x0 (QUERY BLOCK TABLES CHANGED SEL$1)
    SEL$1DBAD500 0x0 (QUERY BLOCK TABLES CHANGED SEL$F486F43F)|
    SEL$6D7A4A3D 0x0 (SPLIT/MERGE QUERY BLOCKS SEL$1DBAD500)
  SEL$7D2C682D 0x0 (SPLIT/MERGE QUERY BLOCKS SEL$F486F43F) [FINAL]
    SEL$7A931295 0x0 (PUSHED PREDICATE SEL$7D2C682D; SEL$7C398D44; "VW_GBC_5"@"SEL$F486F43F" 1)

:
call(in-use=85904, alloc=114672), compile(in-use=364968, alloc=472468), execution(in-use=498372, alloc=499124)

For Help, press F1
```



ask  
Tom

11gr2\_train1 x train1\_ora10 x EVENTS\_LARGE x

0 seconds

train1\_ora10

```
1 SELECT e.event_no, e.start_date, sum(cost) totcost
2 FROM   events_large e, bookings_large b
3 where  e.event_no = b.event_no
4 group by e.event_no, e.start_date
5
```

# 10G – JOIN before the GROUP BY

Explain Plan x

0 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			31665	843726
HASH		GROUP BY	31665	843726
HASH JOIN			14060	5777591
Access Predicates				
E.EVENT_NO=B.EVENT_NO				
TABLE ACCESS	EVENTS_LARGE	FULL	225	99827
TABLE ACCESS	BOOKINGS_LARGE	FULL	8044	5777591



ask  
Tom

11gr2\_train1 x train1\_oracle x EVENTS\_LARGE x

0.016 seconds

11gr2\_train1

```
1 SELECT e.event_no, e.start_date, sum(cost) totcost
2 FROM   events_large e, bookings_large b
3 where  e.event_no = b.event_no
4 group by e.event_no, e.start_date
5
6
```

11G – GROUP BY before the JOIN

Query Result x Query Result 1 x Query Result 2 x Query Result 3 x Explain Plan x

0.016 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			19036	100322
HASH		GROUP BY	19036	100322
HASH JOIN			18176	100322
Access Predicates				
E.EVENT_NO=ITEM_1				
TABLE ACCESS	EVENTS_LARGE	FULL	273	100322
VIEW	VW_GBC_5		17637	100792
HASH		GROUP BY	17637	100792
TABLE ACCESS	BOOKINGS_LARGE	FULL	9836	5767168

Document - WordPad

File Edit View Insert Format Help

SQL Monitoring Report

SQL Text

-----

select /\*+ MONITOR \*/ count(comments) from bookings

Global Information

-----

Status

: DONE (ALL ROWS)

Instance ID

: 1

Session

: TRAIN1 (64:612)

SQL ID

: 6hxa9n8nytp4u

SQL Execution ID

: 16777217

Execution Started

: 08/09/2010 15:52:43

First Refresh Time

: 08/09/2010 15:52:43

Last Refresh Time

: 08/09/2010 15:52:43

Duration

: .000169s

Module/Action

: SQL Developer/-

Service

: SYS\$USERS

Program

: SQL Developer

Fetch Calls

: 1

Global Stats

=====

Elapsed	Other	Fetch	Buffer
Time(s)	Waits(s)	Calls	Gets
=====		=====	
0.00	0.00	1	7
=====		=====	

SQL Plan Monitoring Details (Plan Hash Value=21997667)

=====

Id	Operation	Name	Rows	Cost	Time	Start	Execs	Rows	Activity	Activity Detail
			(Estim)		Active(s)	Active		(Actual)	(%)	(# samples)
=====		=====		=====		=====		=====		=====
0	SELECT STATEMENT				1	+0	1	1		
1	SORT AGGREGATE		1		1	+0	1	1		
2	TABLE ACCESS FULL	BOOKINGS	22	3	1	+0	1	22		
=====		=====		=====		=====		=====		=====

For Help, press F1

# Upgrading

Check the differences in plans

```

rundiff.txt - Notepad
File Edit Format View Help

set serveroutput on
set define off

declare
  result varchar2(1000);
BEGIN
  result := dbms_xplan.diff_plan_outline(
    sql_text      => 'SELECT e.event_no, e.start_date FROM events_large e WHERE event_no
NOT IN
(SELECT event_no FROM bookings_large WHERE status = 'P' AND cost > 100)',
    outline1      => 'OPTIMIZER_FEATURES_ENABLE('11.1.0.7')',
    outline2      => 'OPTIMIZER_FEATURES_ENABLE('10.1.0.2')');
  dbms_output.put_line('diff is '||result);
end;

URL:|
http://host.my.com:portnumber/orarep/plandiff/all?task_id=238&format=html&method=qbreg

select dbms_report.get_report(
  '/orarep/plandiff/all?task_id=238&format=text&method=qbreg')
from dual;

```



```
DBMS_REPORT.GET_REPORT('/ORAREP/PLANDIFF/ALL?TASK_ID=238&FORMAT=TEXT&METHOD=QBRE
```

```
-----
General Information
-----
```

```
-----
Task Information:
```

```
Workload Information:
-----
```

```
Task Name      : TASK_238
Task Owner     : TRAIN1
Description    : diff_plan_outline
```

```
Report Details: SQL Plan Comparison Query Block Diff
-----
```

Query Block	SQL Plan 1	SQL Plan 2	Diff
.. SEL\$5DA710D3	Yes	NA	.. SUBQUERY UNNEST

```
Plan Id       : 994
Plan Hash Value : 3319067774
```

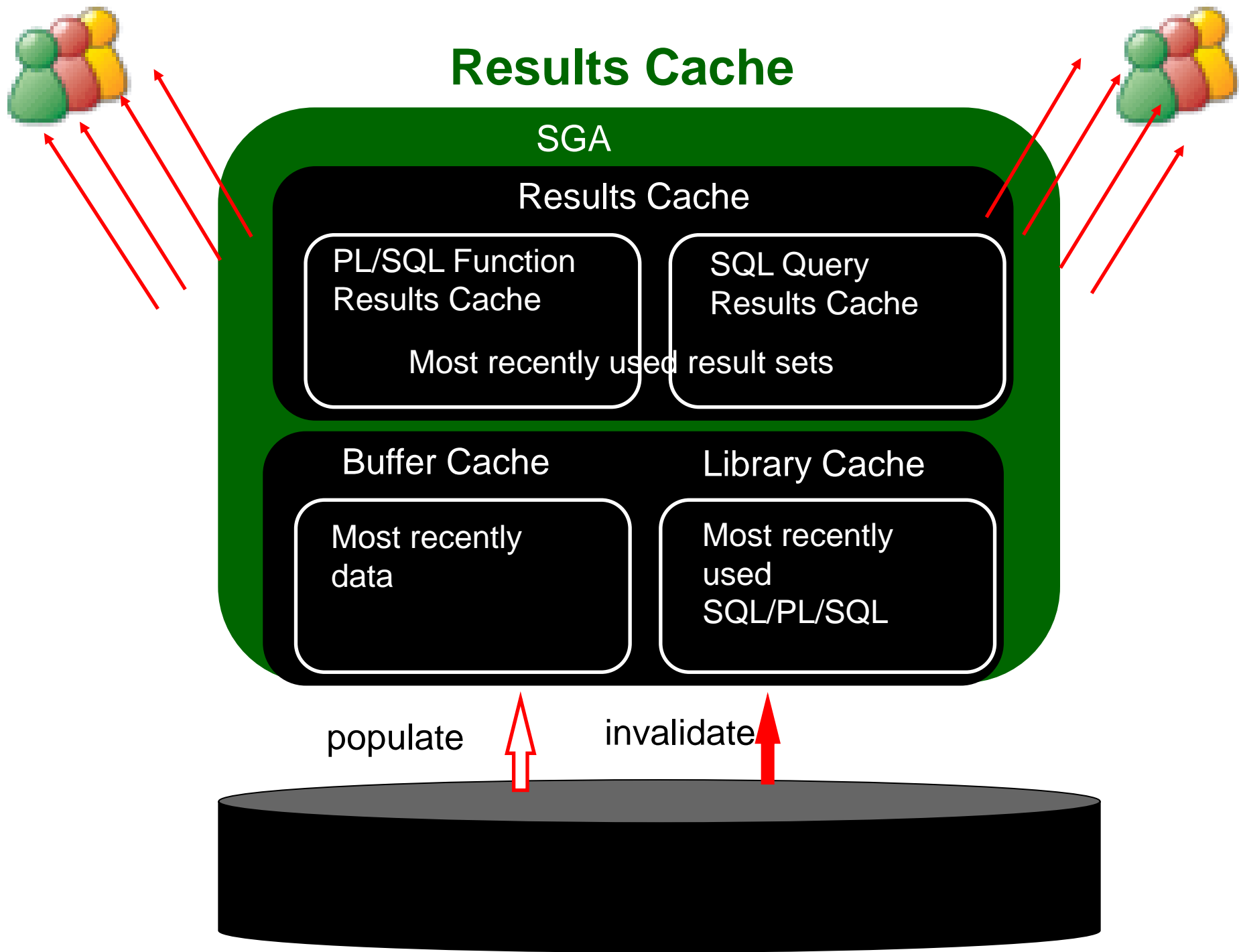
```
-----
| Id | Operation | Name |
-----
```

```
Plan Id       : 995
Plan Hash Value : 1140461147
```

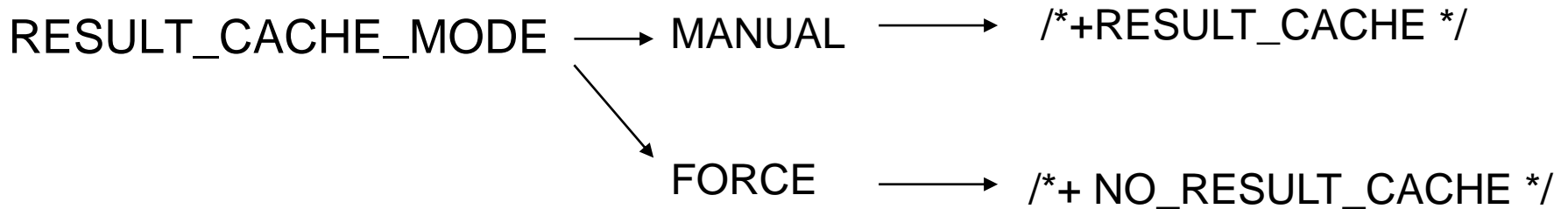
```
-----
| Id | Operation | Name |
-----
```

Diff	Transformation	SQL Plan 1	SQL Plan 2	Query block
*	. PARSER	-	Yes	. SEL\$2
*	.. SUBQUERY UNNEST	Yes	NA	.. SEL\$5DA710D3

Diff	Transformation	SQL Plan 1	SQL Plan 2	Query block
*	. PARSER	-	Yes	. SEL\$1
v	.. SUBQUERY UNNEST	Yes	NA	.. SEL\$5DA710D3



# Query Results Cache



```
SELECT /*+ RESULT_CACHE */  
       count(b.comments)  
FROM train1.events_large e, train1.bookings_large b  
WHERE e.org_id = :v1  
AND   e.event_no = b.event_no  
AND   e.comments = :v2;
```

# PL/SQL Function Results Cache

```
CREATE OR REPLACE FUNCTION quantity_booked
(p_resource_code in resources.code%TYPE,p_event_date in date)
RETURN NUMBER
RESULT_CACHE
IS
    v_total_booked number := 0;
BEGIN
    SELECT          sum(b.quantity)
    INTO  v_total_booked
    FROM    bookings b, events e
    WHERE   e.event_no = b.event_no
    AND     p_event_date between e.start_date and e.end_date
    AND     b.resource_code = p_resource_code;

    RETURN (v_total_booked);
END;
```



# Monitoring the Results Cache

SELECT \* FROM v\$result\_cache\_memory

SELECT \* FROM v\$result\_cache\_objects

SELECT \* FROM v\$result\_cache\_statistics

SELECT \* FROM v\$result\_cache\_dependency

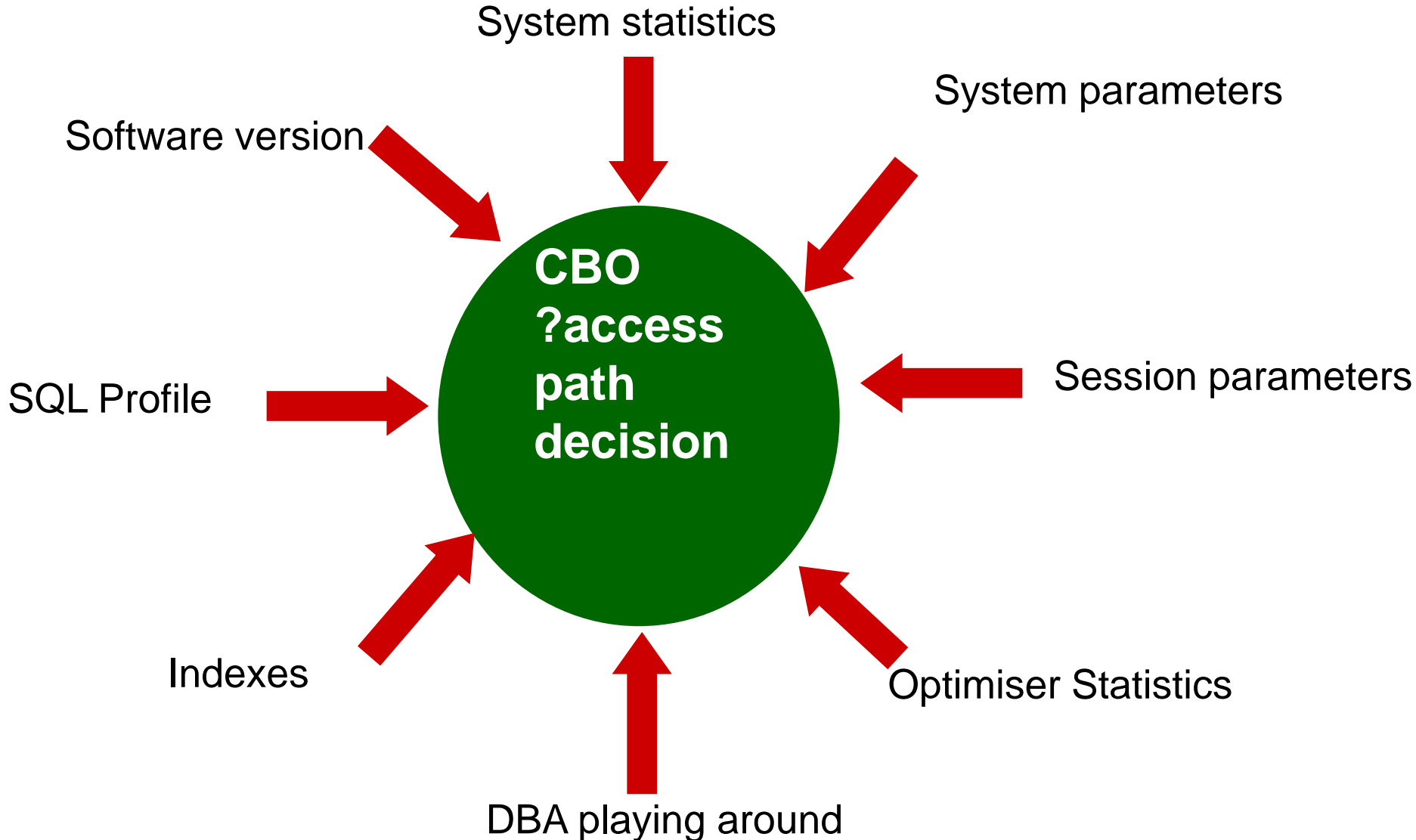
RESULT\_CACHE\_MAX\_SIZE

RESULT\_CACHE\_MAX\_RESULT

Explain Plan

OPERATION	OPTIONS	OBJECT_NAME
SELECT STATEMENT	(null)	(null)
RESULT CACHE	(null)	bzurnr2sm456r4yctszyqks1s5
SORT	AGGREGATE	(null)
NESTED LOOPS	(null)	(null)
TABLE ACCESS	FULL	BOOKINGS_LARGE
INDEX	UNIQUE SCAN	EVTLG_PK

# CBO - Instability



# CBO - Instability

## Solution 1

- Leave it all alone

## Solution 2

- Store plan baseline
- Plans not used till accepted
- Manually accept or
- Allow Oracle to evolve plans

# SQL Plan Management

## Manual capture

DBMS\_SPM.LOAD\_PLANS\_FROM\_SQLSET  
DBMS\_SPM.LOAD\_PLANS\_FROM\_CURSOR\_CACHE

## Auto capture of repeatable statements

OPTIMIZER\_CAPTURE\_SQL\_PLAN\_BASELINE = TRUE

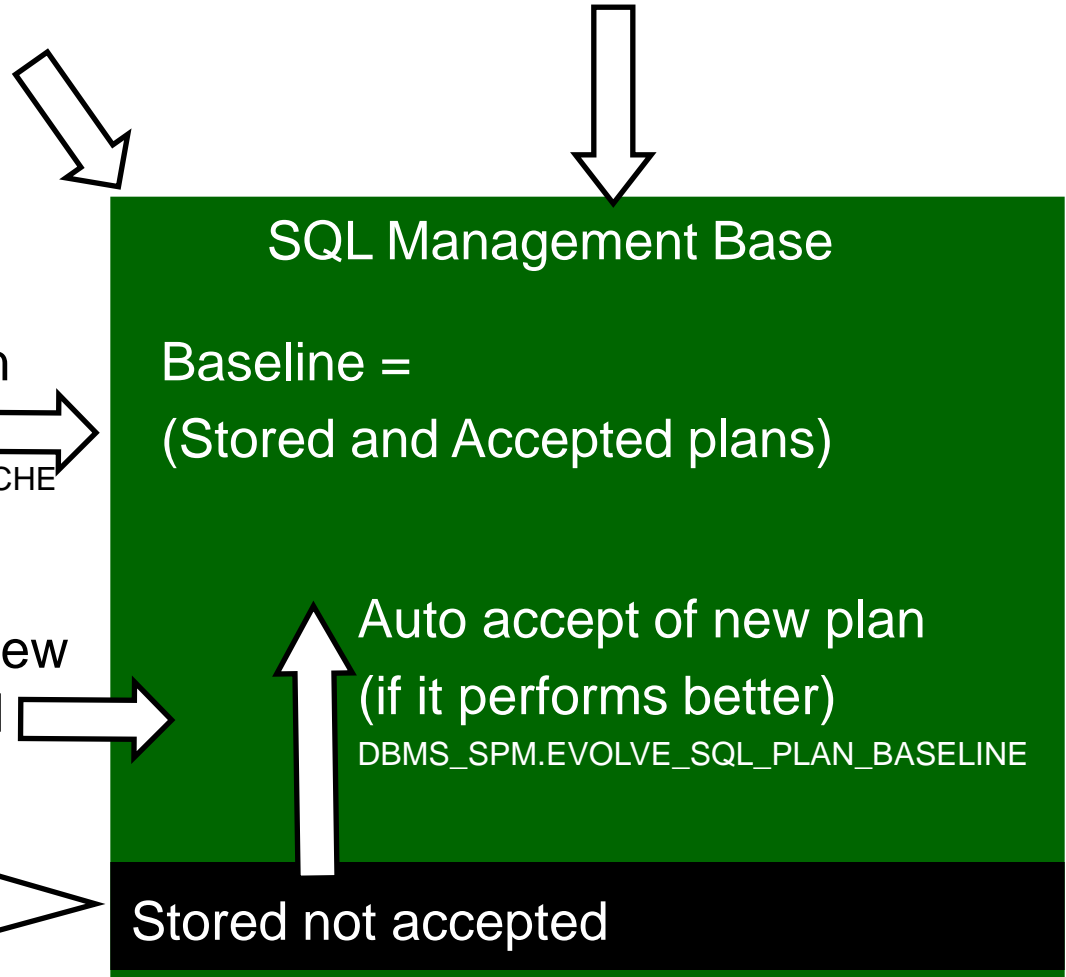
## Manual load/accept of new plan

DBMS\_SPM.LOAD\_PLANS\_FROM\_SQLSET  
DBMS\_SPM.LOAD\_PLANS\_FROM\_CURSOR\_CACHE

SQL Tuning Advisor identifies new  
plan – SQL\*Profile accepted

New Plan

identified during  
execution







# **SAGE Computing Services**

Customised Oracle Training Workshops and Consulting

## **Questions?**

*[www.sagecomputing.com.au](http://www.sagecomputing.com.au)*

*[penny@sagecomputing.com.au](mailto:penny@sagecomputing.com.au)*