Meet the Cost Based Optimiser in 11g

Penny Cookson
SAGE Computing Services

www.sagecomputing.com.au
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
penny@sagecomputing.com.au
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)
Oracle Version < 9

SELECT COUNT(l.quantity) 
FROM bookings_skew l 
WHERE  resource_code = :v1;

'BRLG'

'PC1'

<Version 9 database
No bind peeking
Oracle Version < 9

SELECT COUNT(l.quantity) 
FROM bookings_skew l 
WHERE resource_code = :v1;

‘BRLG’

‘PC1’

<Version 9 database
No bind peeking
What is Bind Peeking?

SELECT COUNT(l.quantity) FROM bookings_skew l WHERE resource_code = :v1;

How many rows do I expect?

>=Version 9 database
Bind peeking

INDEXED
ACCESS

Shared Pool

SELECT COUNT(l.quantity) FROM bookings_skew l WHERE resource_code = :v1;
Histogram – Minority First

SINGLE TABLE ACCESS PATH
Column (#3): RESOURCE_CODE(VARCHAR2)
  AvgLen: 5.00 NDV: 9 Nulls: 0 Density: 9.0892e-008
  Histogram: Freq #Bkts: 9 UncompBkts: 5966 EndPtVals: 9
Table: BOOKINGS_SKEW Alias: L
  Access Path: TableScan
    Cost: 7693.35 Resp: 7693.35 Degree: 0
    Cost_io: 7426.00 Cost_cpu: 1555877511
    Resp_io: 7426.00 Resp_cpu: 1555877511
  Access Path: index (AllEqRange)
    Index: BK_RESSKEW
      resc_io: 2399.00 resc_cpu: 37397785
      ix_sel: 0.0080456 ix_sel_with_filters: 0.0080456
      Cost: 2405.43 Resp: 2405.43 Degree: 1
  Best:: AccessPath: IndexRange Index: BK_RESSKEW
    Cost: 2405.43 Degree: 1 Resp: 2405.43 Card: 43967.55 Bytes: 0

RESO COUNT(*)
---- ----------VCR1 495711CONF 495720LNCH 743576BRSM 743583PC1 47858FLPC 495720BRLG 1739277TAP1 247864VCR2 495715
What is the CBO good at in 10g?

<table>
<thead>
<tr>
<th>Data</th>
<th>Condition</th>
<th>Literal/Bin Var</th>
<th>Histogram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even Distribution</td>
<td>Equality</td>
<td>Literal</td>
<td>N/A</td>
</tr>
<tr>
<td>Even Distribution</td>
<td>Equality</td>
<td>Bind</td>
<td>N/A</td>
</tr>
<tr>
<td>Skewed</td>
<td>Equality</td>
<td>Literal</td>
<td>NO</td>
</tr>
<tr>
<td>Skewed</td>
<td>Equality</td>
<td>Literal</td>
<td>YES</td>
</tr>
<tr>
<td>Skewed</td>
<td>Equality</td>
<td>Bind</td>
<td>NO</td>
</tr>
<tr>
<td>Skewed</td>
<td>Equality</td>
<td>Bind</td>
<td>YES</td>
</tr>
<tr>
<td>Even Distribution</td>
<td>Range</td>
<td>Bind</td>
<td>N/A</td>
</tr>
</tbody>
</table>
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 becomes bind aware.

New values will use a plan for the appropriate selectivity range.

DEMO

Adaptive Cursors
### V$SQL

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>IS_BIND_SENSITIVE</th>
<th>IS_BIND_AWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>7bmrs67hubufy</td>
<td>0 Y</td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>1 Y</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>2 Y</td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>3 Y</td>
<td></td>
<td>Y</td>
</tr>
</tbody>
</table>

### V$SQL_CS_SELECTIVITY

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>SUBSTR(PREDICATE, 1, 10)</th>
<th>RANGE_ID</th>
<th>LOW</th>
<th>HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>7bmrs67hubufy</td>
<td>1 =V1</td>
<td></td>
<td></td>
<td>0.008765</td>
<td>0.010712</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>2 =V1</td>
<td></td>
<td></td>
<td>0.004382</td>
<td>0.010712</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>3 =V1</td>
<td></td>
<td></td>
<td>0.004382</td>
<td>0.096808</td>
</tr>
</tbody>
</table>
### V$SQL_CS_HISTOGRAM

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>BUCKET_ID</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>7bmrs67hubufy</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>3</td>
<td>0</td>
<td>11</td>
</tr>
</tbody>
</table>

### V$SQL_CS_STATISTICS

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>CHILD_NUMBER</th>
<th>BIND_SET_HASH_VALUE</th>
<th>PEEKED</th>
<th>EXECUTIONS</th>
<th>ROWS_PROCESSED</th>
<th>BUFFER_GETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>7bmrs67hubufy</td>
<td>0</td>
<td>2982103524</td>
<td>Y</td>
<td>1</td>
<td>1991053</td>
<td>35333</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>1</td>
<td>1466228028</td>
<td>Y</td>
<td>1</td>
<td>145519</td>
<td>7058</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>2</td>
<td>778730927</td>
<td>Y</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>3</td>
<td>2421911073</td>
<td>N</td>
<td>1</td>
<td>299998</td>
<td>4784</td>
</tr>
<tr>
<td>7bmrs67hubufy</td>
<td>3</td>
<td>3254803217</td>
<td>Y</td>
<td>1</td>
<td>1490902</td>
<td>34471</td>
</tr>
</tbody>
</table>
PLSQL has soft parse avoidance

No default adaptive cursor functionality

Implicit cursor
Explicit cursor
Native dynamic SQL

Default adaptive cursor functionality

Ref cursor
Session_cached_cursors = 0
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_AWARE */
So where are we now?

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

Packages – will still get it wrong once and won’t use adaptive cursors for PL/SQL at all unless we set session cached cursors = 0
SELECT COUNT(l.quantity)
FROM train1.booking_large l
WHERE resource_code = :vl

Details
Select the plan hash value to see the details below. Plan Hash Value: 1457437069
There are multiple plans found for this SQL statement.

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

Detail for Selected 5 Minute Interval
Activity (%), SID, QC_SID, User, Program, Service, Plan Hash Value
100.00, 19, TRAIN1, sqlplus.exe, ora11gr2.sagecomputing.com.au, 1457437069
Select the plan hash value to see the details below.

Plan Hash Value: 1457437069

There are multiple plans found for this SQL statement.

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

Additional Information:

Overview:
- 56,164
- BOOKINGS_LARGE
- 56,164
- SORT AGGREGATE
- 1

Selection Details:
- Nothing Selected
SELECT COUNT(l.quantity)
FROM train1.bookings_large l
WHERE resource_code = :v1
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.

View:

- Overview
- Selection Details
Specify the following parameters to schedule a job to run the SQL Tuning Advisor:

- **Name**: SQL_TUNING_1281147098609
- **Description**: [Field]
- **SQL Tuning Set**: SYS.TOP_SQL_1281147098500
- **SQL Tuning Set Description**: Automatically generated by Top SQL
- **SQL Statements Counts**: 2

### SQL Statements

#### Scope

- **Total Time Limit (minutes)**: 30
- **Scope of Analysis**
  - Limited
  - Comprehensive
  
  The analysis is done without SQL Profile recommendation and takes about 1 second per statement.
  
  This analysis includes SQL Profile recommendation, but may take a long time.

#### Schedule

- **Time Zone**: UTC
- **Immediately**
- **Later**

  - **Date**: 7/08/2010
  
  (example: 7/08/2010)
  
  - **Time**: 10:30
  
  AM PM
Only one recommendation should be implemented.

**SQL Text**

```
SELECT COUNT(*) FROM train1_bookings_large WHERE resource_code = 'v1'
```

**Select Recommendation**

<table>
<thead>
<tr>
<th>Select</th>
<th>Type</th>
<th>Findings</th>
<th>Recommendations</th>
<th>Rationale</th>
<th>Benefit (%)</th>
<th>Other Statistics</th>
<th>New Explain Plan</th>
<th>Compare Explain Plans</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑️</td>
<td>Alternative Plans</td>
<td>Some alternative execution plans for this statement were found by searching the system’s real-time and historical performance data.</td>
<td>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.</td>
<td>Creating a plan baseline for the plan with the best elapsed time will prevent the Oracle optimizer from selecting a plan with worse performance.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Only one recommendation should be implemented.

**SQL Text**

```sql
SELECT COUNT(*) FROM train1 bookings_large I WHERE resource_code = 'v1'
```

**Select Recommendation**

<table>
<thead>
<tr>
<th>Select</th>
<th>Type</th>
<th>Findings</th>
<th>Recommendations</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>Alternative Plans</td>
<td>Some alternative execution plans for this statement were found by searching the system's real-time and historical performance data.</td>
<td>Consider creating a SQL plan baseline for the plan with the best average elapsed time.</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Maybe you should buy an Exadata box**
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
- Manually hack an outline/profile
Adaptive Cursors What we Really Need

Adaptive Cursors Persistence

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

?SQL *Profile to indicate bind aware
Try to get rid of your hints

_set_optimizer_ignore_hints_

Create additional statistics
Set _set_optimizer_ignore_hints to TRUE
Test the app
If it runs OK remove your hints

DEMO
58 New Hints

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

IGNORE_ROW_ON_DUPKEY_INDEX
This has nothing to do with optimisation I just like it

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

from dual;
Much More Query Transformation

```sql
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
```

```sql
alter session set tracefile_identifier = Penny
alter session set events '10053 trace name context forever'
explain plan for
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 events '10053 trace name context off'
```
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
10G – JOIN before the GROUP BY
<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>6668</td>
<td>18.43</td>
<td>22.06</td>
<td>28499</td>
<td>36330</td>
<td>0</td>
<td>100001</td>
</tr>
<tr>
<td>total</td>
<td>6670</td>
<td>18.43</td>
<td>22.06</td>
<td>28499</td>
<td>36330</td>
<td>0</td>
<td>100001</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 87

Rows | Row Source Operation
--- | -------------------
100001 | HASH GROUP BY (cr=36330 pr=28499 pw=0 time=21681647 us)
5767168 | HASH JOIN (cr=36330 pr=28499 pw=0 time=11813763 us)
100322 | TABLE ACCESS FULL EVENTS_LARGE (cr=976 pr=0 pw=0 time=119 us)
5767168 | TABLE ACCESS FULL BOOKINGS_LARGE (cr=35354 pr=28499 pw=0 time=5812854 us)

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>6668</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>db file scattered read</td>
<td>2416</td>
<td>0.04</td>
<td>3.69</td>
</tr>
<tr>
<td>db file sequential read</td>
<td>113</td>
<td>0.02</td>
<td>0.12</td>
</tr>
</tbody>
</table>
11G – GROUP BY before the JOIN
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

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>1</td>
<td>0.01</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>6668</td>
<td>4.03</td>
<td>6.26</td>
<td>35213</td>
<td>36203</td>
<td>0</td>
<td>10000</td>
</tr>
<tr>
<td>total</td>
<td>6670</td>
<td>4.04</td>
<td>6.27</td>
<td>35213</td>
<td>36203</td>
<td>0</td>
<td>10000</td>
</tr>
</tbody>
</table>

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 92

Rows | Row Source          | Operation | (cr=| pr= | pw= | time= | us | cost= |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100000</td>
<td>HASH GROUP BY (cr=36203 pr=35213 pw=0 time=66708 us cost=19036</td>
<td>size=3109982 card=100322</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100000</td>
<td>HASH JOIN (cr=36203 pr=35213 pw=0 time=301403 us cost=18176</td>
<td>size=3109982 card=100322</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100322</td>
<td>TABLE ACCESS FULL EVENTS_LARGE (cr=979 pr=0 pw=0 time=108254 us cost=273 size=1304186 card=100322</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100000</td>
<td>VIEW VW_GBC5 (cr=35224 pr=35213 pw=0 time=128003 us cost=17637</td>
<td>size=1814256 card=100792</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100000</td>
<td>HASH GROUP BY (cr=35224 pr=35213 pw=0 time=57288 us cost=17637</td>
<td>size=806336 card=100792</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5767168</td>
<td>TABLE ACCESS FULL BOOKINGS_LARGE (cr=35224 pr=35213 pw=0 time=7637118 us cost=9836 size=46137344 card=5767168</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Elapsed times include waiting on following events:

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Net message to client</td>
<td>6668</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>direct path read</td>
<td>206</td>
<td>0.06</td>
<td>2.05</td>
</tr>
<tr>
<td>asynch descriptor resize</td>
<td>4</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL Net message from client</td>
<td>6668</td>
<td>15.08</td>
<td>45.14</td>
</tr>
</tbody>
</table>
Registered qbe: SEL$1 0x2a650ac0 (COPY SEL$1)

QUERY BLOCK SIGNATURE

signature(); NULL

Cost-Based Group-By/Distinct Placement

GBP/DP: Checking validity of GBP/DP for query block SEL$1 (#1)
GBP: Checking validity of group-by placement for query block SEL$1 (#1)

GBP: Using search type: exhaustive
GBP: Considering group-by placement on query block SEL$1 (#1)
GBP: Starting iteration 1, state space = (1,2) : (0,0)
GBP: Original query

*** UNPARSED QUERY IS ***

SELECT "E"."EVENT_NO" "EVENT_NO","E"."START_DATE" "START_DATE",SUM("B"."COST") "TOTCOST" FROM "TRAIN1"."EVENTS_LARGE" "E"."TRAIN1"."BOOKINGS_LARGE" "B" WHERE "E"."EVENT_NO"="B"."EVENT_NO" GROUP BY "E"."EVENT_NO","E"."START_DATE" FPD: Considering simple filter push in query block SEL$1 (#1)
"E"."EVENT_NO"="B"."EVENT_NO"
try to generate transitive predicate from check constraints for query block SEL$1 (#1)
finally: "E"."EVENT_NO"="B"."EVENT_NO"

GBP: Costing transformed query.
CBOT: Looking for cost annotations for query block SEL$1, key = SEL$1_00000000_0
CBOT: Could not find stored cost annotations.

kkoqbc: optimizing query block SEL$1 (#1)

: call(in-use=4440, alloc=16360), compile(in-use=141756, alloc=158708), execution(in-use=176260, alloc=179748)

kkoqbc-sbheap (create addr=0x0e425f5c)

**************
QUERY BLOCK TEXT
**************
Not available.

**************
QUERY BLOCK SIGNATURE
**************

signature (optimizer): qbe_name=SEL$1 nbfros=2 flg=0
fro(0): flg=0 objn=74850 hint_alias="B"@"SEL$1"
fro(1): flg=0 objn=74848 hint_alias="E"@"SEL$1"

**************
SYSTEM STATISTICS INFORMATION
**************

Using NOWORKLOAD Stats
CPU SPEDOWN: 1684 millions instructions/sec (default is 100)
TOTFRSPEED: 4096 bytes per millisecond (default is 4096)
IOSLEKTIM: 10 milliseconds (default is 10)
MBRC: -1 blocks (default is 8)

**************
Not available.

QUERY BLOCK SIGNATURE

signature (optimizer): qbnam==SEL$$C308D48 nbrows=2 flg=0
  fro(0): flg=0 o bjno=74848 hint_alias="E" @"SEL$$C"
  fro(1): flg=1 o bjno=0 hint_alias="VW_GBC_5" @"SEL$$F486F43F"

SYSTEM STATISTICS INFORMATION

Using NOWORKLOAD Stats
CPU SPEED: 1684 millions instructions/sec (default is 100)
I/O TRANS: 4996 bytes per millisecond (default is 4096)
IO SEEK TIME: 10 milliseconds (default is 10)
MBRC: -1 blocks (default is 8)

BASE STATISTICAL INFORMATION

Table Stats:
  Table: EVENTS_LARGE Alias: E
    #Rows: 100322 #Blks: 1000 AvgRowLen: 64.00
  Index Stats:
    Index: EVTL_S_PK col# 1
      LVL: 1 #L: 188 #D: 100322 LB/K: 1.00 DB/K: 1.00 CLUF: 956.00

Table Stats:
  Table: VW_GBC_5 Alias: VW_GBC_5 NO STATISTICS
  Column (#1): ITEM_ID
    AvgLen: 5 NOV: 100792 Nulls: 0 Density: 0.000010 Min: 201 Max: 100200

Access path analysis for VW_GBC_5

SINGLE TABLE ACCESS PATH
Single Table Cardinality Estimation for EVENTS_LARGE[E]
  Table: EVENTS_LARGE Alias: E
  Card: Original: 100322.000000 Rounded: 100322 Computed: 100322.00 Non Adjusted: 100322.00
  Access Path: TableScan
    Cost: 273.49 Resp: 273.49 Degree: 0
    Cost_10: 272.00 Cost_cpu: 30195500
    Resp_10: 272.00 Resp_cpu: 30195500
  Best: AccessPath: TableScan
    Cost: 273.49 Degree: 1 Resp: 273.49 Card: 100322.00 Bytes: 0

Grouping column cardinality [ITEM_ID] 100322
Grouping column cardinality [START_DATE] 12

OPTIMIZER STATISTICS AND COMPUTATIONS
fix 8355120 = enabled
fix 7176746 = enabled
fix 8442891 = enabled
fix 8323216 = enabled
fix 7679164 = enabled
fix 77670533 = enabled
fix 8408665 = enabled
fix 8491999 = enabled
fix 8348392 = enabled
fix 8348585 = enabled
fix 8508056 = enabled
fix 8351778 = enabled
fix 8515269 = enabled
fix 8247017 = enabled
fix 7325597 = enabled
fix 8531490 = enabled
fix 6163600 = enabled
fix 8569278 = disabled
fix 8557992 = enabled
fix 7559098 = enabled
fix 8580883 = enabled
fix 5862589 = disabled
fix 8569714 = enabled
fix 8514561 = enabled
fix 8619631 = disabled

Query Block Registry:
SEL$1 0x0 (PARSER)
SEL$54B3641 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST LIST 3)
SEL$54B3641 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST LIST 3)
SEL$706658FA 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST 2)
SEL$38F5D958 0x0 (QUERY BLOCK TABLES CHANGED SEL$1)
SEL$137A03FC 0x0 (SPLIT/MERGE QUERY BLOCKS SEL$38F5D958)
SEL$7C398D44 0x0 (UNKNOWN QUERY BLOCK ORIGIN SEL$1; SEL$1; LIST 5) [FINAL]
SEL$5931295 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$1DB0500 0x0 (QUERY BLOCK TABLES CHANGED SEL$F486F43F)
SEL$6D7A4A3D 0x0 (SPLIT/MERGE QUERY BLOCKS SEL$1DB0500)
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)

End of Optimizer State Dump
Dumping Hints

================================= END SQL Statement Dump ===========================
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
----- Explain Plan Dump -----
----- Plan Table -----

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH GROUP BY</td>
<td></td>
<td>98K</td>
<td>3037K</td>
<td>19K</td>
<td>00:04:49</td>
</tr>
<tr>
<td>2</td>
<td>HASH JOIN</td>
<td></td>
<td>98K</td>
<td>3037K</td>
<td>18K</td>
<td>00:04:39</td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS FULL</td>
<td>EVENTS_LARGE</td>
<td>98K</td>
<td>1772K</td>
<td>17K</td>
<td>00:04:32</td>
</tr>
<tr>
<td>4</td>
<td>VIEW</td>
<td>VW_GBC_3</td>
<td>98K</td>
<td>787K</td>
<td>273</td>
<td>00:04:04</td>
</tr>
<tr>
<td>5</td>
<td>HASH GROUP BY</td>
<td></td>
<td>98K</td>
<td>787K</td>
<td>273</td>
<td>00:04:32</td>
</tr>
<tr>
<td>6</td>
<td>TABLE ACCESS FULL</td>
<td>BOOKINGS_LARGE</td>
<td>5632K</td>
<td>44M</td>
<td>9836</td>
<td>00:02:59</td>
</tr>
</tbody>
</table>

Predicate Information:

1 - access("E."EVENT_NO"="ITEM_1")

Content of other xml column:

db_version: 11.2.0.1
parse_schema: TRAIN1
plan_hash: 1200697483
plan_hash_2: 3556861701
Outline Data:

```
BEGIN_OUTLINE_DATA
  IGNORE_OPTIM_EMBEDDED_HINTS
  OPTIMIZER_FEATURES_ENABLE('11.2.0.1')
  DB_VERSION('11.2.0.1')
  OPT_PARAM('optimizer_index_cost_adj' 20)
  ALL_ROWS
  OUTLINE_LEAF(@"SEL\$7d2c682d")
  OUTLINE_LEAF(@"SEL\$7c98d44")
  PLACE_GROUP_BY(@"SEL\$1" ("B"@"SEL\$1") 5)
  OUTLINE(@"SEL\$f486f43f")
  OUTLINE(@"SEL\$1")
  FULL(@"SEL\$7c98d44" "E"@"SEL\$1")
  NO_ACCESS(@"SEL\$7c98d44" "VW_GBC_5"@"SEL\$f486f43f")
  LEADING(@"SEL\$7c98d44" "E"@"SEL\$1" "VW_GBC_5"@"SEL\$f486f43f")
  USE_HASH(@"SEL\$7c98d44" "VW_GBC_5"@"SEL\$f486f43f")
  USE_HASH_AGGREGATION(@"SEL\$7c98d44")
  FULL(@"SEL\$7d2c682d" "B"@"SEL\$1")
  USE_HASH_AGGREGATION(@"SEL\$/d2c682d")
END_OUTLINE_DATA
```
10G – NOT IN (PROMISE_NO NULLABLE)

```sql
SELECT count(*)
FROM election_promises p
WHERE p.promise_no NOT IN
(SELECT promise_no
  FROM fulfilled_promises
  WHERE status = 'Y'
  AND cost <1000000)
```
10G – KILLED AFTER 1 hr

```
20  SELECT count(*)
21  FROM election_promises p
22  WHERE p.promise_no NOT IN
23  (SELECT promise_no
24    FROM fulfilled_promises
25    WHERE status = 'Y'
26    AND cost <1000000)
```
11G – Rewrite as Null Aware Anti join

```
SELECT count(*)
FROM election_promises p
WHERE p.promise_no NOT IN
(SELECT promise_no
FROM fulfilled_promises
WHERE status = 'Y'
AND cost < 1000000)
```
```
SELECT count(*)
FROM election_promises p
WHERE p.promise_no NOT IN
 |SELECT promise_no
| FROM fulfilled_promises
| WHERE status = 'Y'
| AND cost <1000000|
```

1 row fetched in 2.922 seconds

<table>
<thead>
<tr>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>322</td>
</tr>
</tbody>
</table>
10G – Full Outer Join – default behaviour

```sql
SELECT count(s.seat_no), count(s.description), count(g.preferences)
FROM seat s FULL OUTER JOIN green_votes g ON (s.seat_no = g.seat_no)
```
SELECT count(s.seat_no), count(s.description), count(g.preferences)
FROM seat s FULL OUTER JOIN green_votes g ON (s.seat_no = g.seat_no)
SELECT /*+ NATIVE_FULL_OUTER_JOIN */
    count(e.comments) nume,
    count (b.comments) numb
FROM   events_large e
    FULL OUTER JOIN  bookings_large b
ON (e.event_no = b.event_no)
11G Native Full Outer Join

```
SELECT count(s.seat_no), count(s.description), count(g.preferences)
FROM seat s FULL OUTER JOIN green_votes g ON (s.seat_no = g.seat_no)
```
```
1 SELECT count(s.seat_no), count(s.description), count(g.preferences)
2 FROM seat s FULL OUTER JOIN green_votes g ON (s.seat_no = g.seat_no)
```
Check the differences in plans

```
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;
```
<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Diff</th>
<th>Transformation</th>
<th>SQL Plan 1</th>
<th>SQL Plan 2</th>
<th>Query block</th>
</tr>
</thead>
<tbody>
<tr>
<td>994</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>995</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>. PARSER</td>
<td></td>
<td></td>
<td>Yes</td>
<td>. SEL$2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>. SUBQUERY UNNEST</td>
<td>Yes</td>
<td></td>
<td>NA</td>
<td>. SEL$5DA710D3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>. PARSER</td>
<td></td>
<td></td>
<td>Yes</td>
<td>. SEL$1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>. SUBQUERY UNNEST</td>
<td>Yes</td>
<td></td>
<td>NA</td>
<td>. SEL$5DA710D8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
/* SQL Analyze(70,0) */
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)

Details
Select the plan hash value to see the details below. Plan Hash Value 1140461147
There are multiple plans found for this SQL statement.

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

Maximum CPU

Detail for Selected 5 Minute Interval
Start Time 8/08/2010 13:51:34
Activity (%) SID QC SID User Program Service Plan Hash Value
100.00 70 TRAIN1 SQL Developer SYSSUSERS 1140461147
**Oracle 10g – Run a whole load of random statements – all of which require parsing**

### OVERALL TOTALS FOR ALL NON-RECURSIVE STATEMENTS

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>1251</td>
<td>1.87</td>
<td>1.90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>1252</td>
<td>0.04</td>
<td>0.05</td>
<td>0</td>
<td>57</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Fetch</td>
<td>2700</td>
<td>25.32</td>
<td>25.38</td>
<td>0</td>
<td>429131</td>
<td>0</td>
<td>12400</td>
</tr>
<tr>
<td>total</td>
<td>5203</td>
<td>27.25</td>
<td>27.34</td>
<td>0</td>
<td>429188</td>
<td>0</td>
<td>12402</td>
</tr>
</tbody>
</table>

- Misses in library cache during parse: 1251
- Misses in library cache during execute: 1

**Elapsed times include waiting on following events:**

<table>
<thead>
<tr>
<th>Event waited on</th>
<th>Times Waited</th>
<th>Max. Wait</th>
<th>Total Waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL*Net message to client</td>
<td>4251</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>SQL*Net message from client</td>
<td>4251</td>
<td>0.04</td>
<td>1.48</td>
</tr>
<tr>
<td>SQL*Net break/reset to client</td>
<td>198</td>
<td>0.00</td>
<td>0.01</td>
</tr>
</tbody>
</table>

### OVERALL TOTALS FOR ALL RECURSIVE STATEMENTS

<table>
<thead>
<tr>
<th>call</th>
<th>count</th>
<th>cpu</th>
<th>elapsed</th>
<th>disk</th>
<th>query</th>
<th>current</th>
<th>rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parse</td>
<td>65</td>
<td>0.01</td>
<td>0.01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Execute</td>
<td>165</td>
<td>0.04</td>
<td>0.02</td>
<td>0</td>
<td>4363</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fetch</td>
<td>298</td>
<td>0.04</td>
<td>0.02</td>
<td>0</td>
<td>4363</td>
<td>0</td>
<td>220</td>
</tr>
<tr>
<td>total</td>
<td>526</td>
<td>0.06</td>
<td>0.04</td>
<td>0</td>
<td>4363</td>
<td>0</td>
<td>220</td>
</tr>
</tbody>
</table>

- Misses in library cache during parse: 15
- Misses in library cache during execute: 15
Oracle 11g – Run a whole load of random statements – all of which require parsing

So tuning the Shared Pool becomes more important, but overall the 11g CBO is pretty smart
Results Cache

SGA Results Cache
- PL/SQL Function Results Cache
  - Most recently used result sets
- SQL Query Results Cache
  - Most recently used result sets

Buffer Cache
- Most recently used data

Library Cache
- Most recently used SQL/PL/SQL

populate

invalidate
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;
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

DEMO

RESULT_CACHE_MAX_RESULT

Explain Plan

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>OPTIONS</th>
<th>OBJECT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>(null)</td>
<td>(null)</td>
</tr>
<tr>
<td>RESULT CACHE</td>
<td>(null)</td>
<td>bzurnr2sm456r4yctszyqks1s5</td>
</tr>
<tr>
<td>SORT</td>
<td>AGGREGATE</td>
<td>(null)</td>
</tr>
<tr>
<td>NESTED LOOPS</td>
<td>(null)</td>
<td>(null)</td>
</tr>
<tr>
<td>TABLE ACCESS</td>
<td>FULL</td>
<td>BOOKINGS_LARGE</td>
</tr>
<tr>
<td>INDEX</td>
<td>UNIQUE SCAN</td>
<td>EVTTLG_PK</td>
</tr>
</tbody>
</table>
Setting Statistics Gathering Defaults

Obsolete:

GET_PARAM
SET_PARAM Procedure

Use:

GET_PREFS Function
SET_TABLE_PREFS (set for individual tables)
SET_DATABASE_PREFS (sets for all tables, can include/exclude SYS tables)
SET_GLOBAL_PREFS – for new objects
Gathering Statistics

Early CBO: “Make sure you gather statistics regularly”

Later CBO: “Don’t gather statistics unless data patterns change”

BUT

If you have new majority values you need to recreate the histogram
Gathering and Publishing

Set preferences to `PUBLISH = 'FALSE'`

Gather Statistics

- `user_tab_pending_stats`
- `user_ind_pending_stats`
- `user_col_pending_stats`

Run test case

Alter session set

- `optimizer_pending_statistics = TRUE`

Better

- `dbms_stats.publish_pending_stats`

Worse

- `dbms_stats.delete_pending_stats`
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
CBO - Instability

- System statistics
- System parameters
- Session parameters
- Optimiser Statistics
- Indexes
- DBA playing around
- SQL Profile
- Software version

?access path decision
CBO - Instability

Solution 1
- Leave it all alone

Solution 2
- SQL Plan Management
- 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

SQL Management Base
Baseline =
(Stored and Accepted plans)

Auto accept of new plan
(if it performs better)
DBMS_SPM.EVOLVE_SQL_PLAN_BASELINE

Stored not accepted

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
penny@sagecomputing.com.au