
An Introduction to Cost Based Optimization

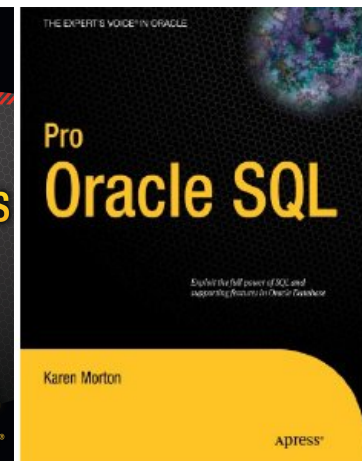
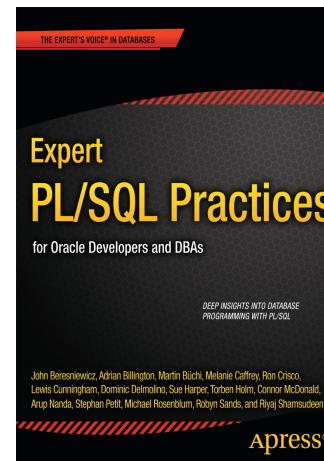
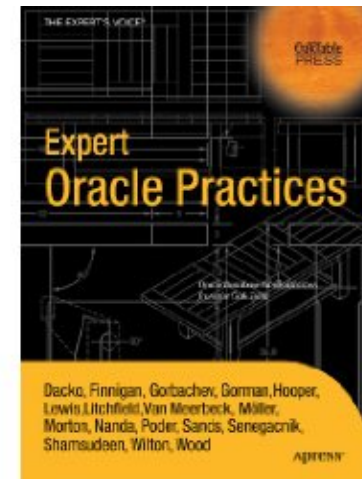
By
Riyaj Shamsudeen



Me



- 18 years using Oracle products/DBA
- OakTable member
- Oracle ACE
- Certified DBA versions 7.0,7.3,8,8i,9i & 10g
- Specializes in RAC, performance tuning, Internals and E-business suite
- Chief DBA with OraInternals
- Email: rshamsud@orainternals.com
- Blog : orainternals.wordpress.com
- URL: www.orainternals.com



Disclaimer

These slides and materials represent the work and opinions of the author and do not constitute official positions of my current or past employer or any other organization. This material has been peer reviewed, but author assume no responsibility whatsoever for the test cases.

If you corrupt your databases by running my scripts, you are solely responsible for that.

This material should not be reproduced or used without the authors' written permission.

Agenda

- **Selectivity**
- Cardinality & histograms
- Index based access cost
- Correlation issues

Selectivity

Selectivity is a measure, with values ranging between 0 and 1, indicating how many elements will be selected from a fixed set of elements.

Selectivity – Example

- Selectivity of predicate

dept =:b1

is $1/\text{NDV}=1/3=0.333$

- Selectivity of predicate

(dept=:b1 or code=:b2)

$= \sim (1/3) + (1/3) - (1/3)*(1/3)$

$= 0.555$

- Selectivity of predicate

(dept=:b1 and code=:b2)

$= \sim (1/3)*(1/3)$

$= 0.111$

Dept	Code	Emp
10	TX	Scott
10	TX	Mary
10	TX	Larry
20	CA	Juan
20	CA	Raj
20	CA	Pele
20	CA	Ronaldinho
30	FL	Ronaldo

NDV : # of distinct values

Demo: demo_01.sql demo_02.sql

Selectivity

- Lower the selectivity, better the predicates are.
- Probability concepts will be applied in calculating selectivity of many predicates.

$$p(A \text{ and } B) = p(A) * p(B)$$

$$p(A \text{ or } B) = p(A) + p(B) - p(A) * p(B)$$

Agenda - Part I

- Selectivity
- **Cardinality & histograms**
- Index cost
- Correlation issues

Cardinality

- Cardinality is a measure, an estimate of number of rows expected from a row source.

- In a very simple sense:

$$\text{Cardinality} = \text{Selectivity} * \text{Number of rows}$$

- Cardinality estimates are very essential and need to be accurate for the optimizer to choose optimal execution plan.

Cardinality – No histogram

- Cardinality of predicate

dept =:b1

= (Selectivity * Num_rows)

= (1/3) * 8 = 2.6 rows

= 3 rows

predicate	Estimate	Actual	Change
dept=10	3	3	=
dept=20	3	4	↑
dept=30	3	1	↓

Dept	Emp
10	Scott
10	Mary
10	Larry
20	Karen
20	Jill
20	Pele
20	Ronaldinho
30	Ronaldo

- Under estimation of cardinality is a root cause for many SQL performance issues.

List

- Calculation of selectivity for an in list is sum of selectivity of individual elements in the list.

```
explain plan for select * from tlist where n3 in (10, 20, 30,40);
```

```
-----  
| Id | Operation          | Name | Rows | Bytes | Cost (%CPU)| Time      |  
-----  
|  0 | SELECT STATEMENT   |      |  400 |  5200 |  104 (2)| 00:00:02 |  
|*  1 | TABLE ACCESS FULL| TLIST|  400 |  5200 |  104 (2)| 00:00:02 |  
-----
```

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

```
-----  
1 - filter("N3"=10 OR "N3"=20 OR "N3"=30 OR "N3"=40)
```

- Individual selectivity of $n3 = \text{predicate}$ is $1/1000$.
- Cardinality = $100,000 \times 4 \times (1/1000) = 400$
- Duplicates eliminated for literal variables

Demo: demo_03.sql demo_03a.sql demo_03b.sql

Range

- Selectivity of range predicate is calculated as
$$= \text{selected_range} / \text{total range} + \text{selectivity of range end pts}$$

explain plan for select * from tlist where n3 between 100 and 200;

```
-----  
| Id | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |  
-----  
|  0 | SELECT STATEMENT  |      | 10210 | 129K | 103 (1) | 00:00:02 |  
|*  1 | TABLE ACCESS FULL| TLIST| 10210 | 129K | 103 (1) | 00:00:02 |  
-----
```

Predicate Information (identified by operation id):

```
-----  
1 - filter("N3"<=200 AND "N3">=100)
```

- Total range : Max - Min = 999 -0

Selected range: 200-100

Cardinality = $100,000 * (2 * (1/1000) + (200-100)/(999-0))$

Demo: demo_04.sql

Range with bind

- If the bind variables are used then the estimate is approximated to 0.25%

```
explain plan for select * from tlist where n3 between :b1 and :b2;
```

```
-----
```

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		250	3250	104 (2)	00:00:02
* 1	FILTER					
* 2	TABLE ACCESS FULL	TLIST	250	3250	104 (2)	00:00:02

```
-----
```

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

- ```

```
- 1 - filter(TO\_NUMBER(:B1)<=TO\_NUMBER(:B2))
  - 2 - filter("N3">=TO\_NUMBER(:B1) AND "N3"<=TO\_NUMBER(:B2))

---

Demo: demo\_05.sql

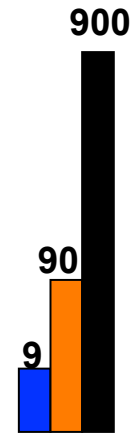
# Cardinality & Skew

```
drop table t1;
create table t1 (color_id number, color varchar2(10));

insert into t1
select l1,
 case
 when l1 <10 then 'blue'
 when l1 between 10 and 99 then 'red'
 when l1 between 100 and 999 then 'black'
 end case
from (select level l1 from dual connect by level <1000)
/

commit;

Begin
 dbms_stats.gather_table_stats ('cbo2','t1',
 estimate_percent =>100, method_opt =>' for all columns size 1');
End;
/
```



# Cardinality : w/o histograms

```
select color, count(*) from t1 group by color order by 2
```

```
COLOR COUNT(*)
```

```

blue 9
red 90
white 900
```

```
select count(*) from t1 where color='blue';
```

```
SQL> select * from table (dbms_xplan.display_cursor);
```

```

| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |

0	SELECT STATEMENT				3 (100)	
1	SORT AGGREGATE		1	6		
* 2	TABLE ACCESS FULL	T1	333	1998	3 (0)	00:01

```

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

```

2 - filter("COLOR"='blue')
```

---

# Histograms

**If the column values have skew and if there are predicates accessing that column, then use histograms. But, use them sparingly!**

# Cardinality : with histograms

Begin

```
dbms_stats.gather_table_stats ('cbo2','t1', estimate_percent =>100,
method_opt =>' for all columns size 10');
end;
/
```

PL/SQL procedure successfully completed.

```
select count(*) from t1 where color='blue';
SQL> select * from table (dbms_xplan.display);
```

| Id  | Operation         | Name | Rows | Bytes | Cost (%CPU) | Time     |
|-----|-------------------|------|------|-------|-------------|----------|
| 0   | SELECT STATEMENT  |      | 1    | 6     | 3 (0)       | 00:00:01 |
| 1   | SORT AGGREGATE    |      | 1    | 6     |             |          |
| * 2 | TABLE ACCESS FULL | T1   | 9    | 54    | 3 (0)       | 00:00:01 |

Predicate Information (identified by operation id):

```
2 - filter("COLOR"='blue')
```

14 rows selected.

---

# Histograms ...

- If there are few distinct values then use that many # of buckets. Else use size auto.
- In many cases, 100% statistics might be needed.
- Collecting histograms in all columns is not recommended and has a side effect of increasing CPU usage. Also, statistics collection can run longer.

---

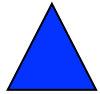
## Agenda - Part I

- Selectivity
- Cardinality & histograms
- Index base access
- **Correlation issues**

---

# Correlation explained

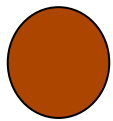
Optimizer assumes no correlation between predicates.



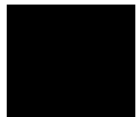
In this example,

All triangles are blue.

All circles are red.



All squares are black.



Predicates shape='CIRCLE' and color='RED'  
are correlated.

But optimizer assumes no-correlation.

# Correlation ...

```
drop table t1;
create table t1
 (color_id number, color varchar2(10), shape varchar2(10));

insert into t1
select l1,
 case
 when l1 <10 then 'blue'
 when l1 between 10 and 99 then 'red'
 when l1 between 100 and 999 then 'black'
 end case,
 case
 when l1 <10 then 'triangle'
 when l1 between 10 and 99 then 'circle'
 when l1 between 100 and 999 then 'rectangle'
 end case
from
 (select level l1 from dual connect by level <1000)
/

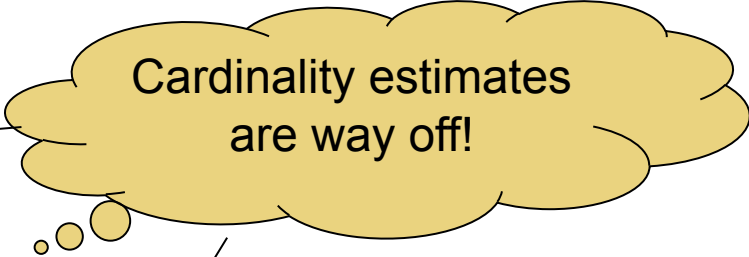
commit;

exec dbms_stats.gather_table_stats ('cbo2','t1', estimate_percent =>100,
 method_opt =>' for all columns size 1');
```

# Correlation & cardinality

```
1* select color, shape, count(*) from t1 group by color,shape
SQL> /
```

| COLOR | SHAPE     | COUNT(*) |
|-------|-----------|----------|
| blue  | triangle  | 9        |
| red   | circle    | 90       |
| black | rectangle | 900      |



Cardinality estimates  
are way off!

```
explain plan for select count(*) from t1
 where color='blue' and shape='triangle';
```

```
select * from table(dbms_xplan.display);
```

| Id  | Operation         | Name | Rows | Bytes | Cost (%CPU) | Time    |
|-----|-------------------|------|------|-------|-------------|---------|
| 0   | SELECT STATEMENT  |      | 1    | 16    | 3 (0)       | 0:00:01 |
| 1   | SORT AGGREGATE    |      | 1    | 16    |             |         |
| * 2 | TABLE ACCESS FULL | T1   | 111  | 1776  | 3 (0)       | 0:00:01 |

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

```
2 - filter("COLOR"='blue' AND "SHAPE"='triangle')
```

## (No)Correlation ..why?

Optimizer assumes no  
Correlation between  
Predicates.

- Selectivity of first single column predicate  
color = 'blue' is 1/3.
- Selectivity of next single column predicate  
shape='triangle' is 1/3.
- Combined selectivity of both predicates are  
 $\text{sel}(p1) * \text{sel}(p2) = (1/3)*(1/3)=1/9$  [ Probability theory ]
- Cardinality estimates, then, becomes  
 $999 * (1/9) = 111$

# Correlation w/ Histograms..

```
alter session set optimizer_dynamic_sampling=0;
```

```
exec dbms_stats.gather_table_stats ('cbo2','t1', estimate_percent =>100,
method_opt =>' for all columns size 5');
```

```
explain plan for select count(*) from t1
where color='blue' and shape='triangle';
```

```
select * from table(dbms_xplan.display);
```

| Id  | Operation         | Name | Rows | Bytes | Cost (%CPU) | Time     |
|-----|-------------------|------|------|-------|-------------|----------|
| 0   | SELECT STATEMENT  |      | 1    | 16    | 3 (0)       | 0:00:01  |
| 1   | SORT AGGREGATE    |      | 1    | 16    |             |          |
| * 2 | TABLE ACCESS FULL | T1   | 1    | 16    | 3 (0)       | 00:00:01 |

With histograms, row  
Estimates are farther away  
from reality

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

```
2 - filter("SHAPE"='triangle' AND "COLOR"='blue')
```

---

## So what do we do?

- Until version 11g, this is a real problem. There is no easy way to fix this. Column statistics might need to be manually adjusted.
- In version 10g, `optimizer_dynamic_sampling` at level 4 can be used to mitigate this.
- Version 11g provides extended statistics to resolve this correlation issue. Refer my blog entry <http://orainternals.wordpress.com/2008/03/21/correlation-between-column-predicates/> for more information on this topic.

---

# Extended statistics

- Dbms\_stats package provides a function to create extended statistics.
- Following code creates an extended statistics on (color,shape) capturing correlation between the columns color and shape.

```
SELECT dbms_stats.create_extended_stats(ownname=>user,
 tabname => 'T1',extension => '(color, shape)') AS c1_c2_correlation
FROM dual;
```

```
C1_C2_CORRELATION
```

```

SYS_STUA0JW6_2K$IUXLR#$DK235BV
```

# Virtual column

- Creating extended statistics adds a hidden, virtual column to that table.

```
select owner, table_name, column_name, hidden_column, virtual_column
from dba_tab_cols where table_name='T1' and owner='CBO2'
order by column_id;
```

| OWNER | TABLE | COLUMN_NAME                      | HID | VIR |
|-------|-------|----------------------------------|-----|-----|
| ...   |       |                                  |     |     |
| CBO2  | T1    | SHAPE                            | NO  | NO  |
| CBO2  | T1    | SYS_STUAOJW6_2K\$IUXLR#\$DK235BV | YES | YES |

- Following code collects statistics on virtual column with histogram.

```
begin
 dbms_stats.gather_Table_stats(user, 'T1',
 estimate_percent => 100,
 method_opt => 'for all columns size 254');
end;
/
```

# Better estimates

...2

- Cardinality estimates are much better.

```
explain plan for select count(*) from t1 where
color='blue' and shape='triangle';
select * from table(dbms_xplan.display);
```

| Id  | Operation         | Name | Rows | Bytes | Cost (%CPU) | Time     |
|-----|-------------------|------|------|-------|-------------|----------|
| 0   | SELECT STATEMENT  |      | 1    | 16    | 4 (0)       | 00:00:01 |
| 1   | SORT AGGREGATE    |      | 1    | 16    |             |          |
| * 2 | TABLE ACCESS FULL | T1   | 9    | 144   | 4 (0)       | 00:00:01 |

Predicate Information (identified by operation id):

```
2 - filter("SHAPE"='triangle' AND "COLOR"='blue')
```

# Better estimates

- For other combinations also estimate is very close.

```
SQL> explain plan for select count(*) from t1 where color='black' and
 shape='rectangle';
```

Explained.

```
SQL> select * from table(dbms_xplan.display);
```

| Id  | Operation         | Name | Rows | Bytes | Cost (%CPU) | Time     |
|-----|-------------------|------|------|-------|-------------|----------|
| 0   | SELECT STATEMENT  |      | 1    | 16    | 4 (0)       | 00:00:01 |
| 1   | SORT AGGREGATE    |      | 1    | 16    |             |          |
| * 2 | TABLE ACCESS FULL | T1   | 900  | 14400 | 4 (0)       | 00:00:01 |

Predicate Information (identified by operation id):

```
2 - filter("COLOR"='black' AND "SHAPE"='rectangle')
```

# Non-existing values

- For non-existing combinations also, estimates are much better.

```
SQL> explain plan for select count(*) from t1 where color='blue' and
 shape='rectangle';
```

Explained.

```
SQL> select * from table(dbms_xplan.display);
```

| Id  | Operation         | Name | Rows | Bytes | Cost (%CPU) | Time     |
|-----|-------------------|------|------|-------|-------------|----------|
| 0   | SELECT STATEMENT  |      | 1    | 16    | 4 (0)       | 00:00:01 |
| 1   | SORT AGGREGATE    |      | 1    | 16    |             |          |
| * 2 | TABLE ACCESS FULL | T1   | 5    | 80    | 4 (0)       | 00:00:01 |

Predicate Information (identified by operation id):

```
2 - filter("COLOR"='blue' AND "SHAPE"='rectangle')
```

---

# References

1. Oracle support site. Metalink.oracle.com. Various documents
2. Internal's guru Steve Adam's website  
[www.ixora.com.au](http://www.ixora.com.au)
3. Jonathan Lewis' website  
[www.jlcomp.daemon.co.uk](http://www.jlcomp.daemon.co.uk)
4. Julian Dyke's website  
[www.julian-dyke.com](http://www.julian-dyke.com)
5. 'Oracle8i Internal Services for Waits, Latches, Locks, and Memory'  
by Steve Adams
6. Randolph Geist : <http://oracle-randolf.blogspot.com>
7. Tom Kyte's website  
[Asktom.oracle.com](http://asktom.oracle.com)