Effective use of column histograms

Paul Baumgartel
Charles Island Consulting
NYOUG, March 15, 2007

What are histograms?

- Data dictionary objects
- Document the distribution of values in a column
- Created by DBMS_STATS
- If used, must be kept up to date

© 2007 Paul Baumgartel

Why histograms?

- Oracle's optimizer relies heavily on *cardinality* in deciding on execution plans
- Cardinality is derived from *selectivity*
- Selectivity is the fraction of rows the optimizer expects to find, based on SQL predicate

© 2007 Paul Baumgartel

Why histograms? continued

- Cardinality is computed as the number of input rows multiplied by selectivity
- Another way to express cardinality is the number of rows expected to result from a given predicate applied to a given row source

© 2007 Paul Baumgartel

Why histograms? continued

- DBMS_STATS by default calculates
  - The number of rows in a table
  - The number of distinct values in a column
- Without additional information, optimizer assumes that values are evenly distributed
- But if they're *not* evenly distributed, inefficient execution plans can result

© 2007 Paul Baumgartel

How the optimizer uses histograms

- It uses information about value distribution in creating execution plan
- Histograms can be useful for both indexed and non-indexed columns
- The two most common uses are
  - Determining whether to use an indexed access path
  - Determining join order

© 2007 Paul Baumgartel

Without a histogram: example

- Consider a table of 1 million rows
- An indexed column contains 500K distinct values
- A query contains an equality predicate on the indexed column
- Values are not evenly distributed
  - 400K rows contain one value
  - The other 499,999 values occur in the other 600K rows

© 2007 Paul Baumgartel

Without a histogram: example

- Selectivity = 1/500,000 = 0.000002
- Cardinality = 0.000002 * 1,000,000 = 2
- Number of rows expected = 2
- Optimizer will choose indexed access path
- Perfectly reasonable if the value in the predicate is not the “popular” value

© 2007 Paul Baumgartel

Without a histogram: example

- Suppose the value in the predicate is the "popular" value
- Actual selectivity is 400,000 / 1,000,000 = 0.4
- Actual cardinality is 0.4 * 1,000,000 = 400,000
- When retrieving 400,000 out of 1,000,000 rows, a full table scan is less expensive than indexed
- But optimizer doesn't know this, and chooses to use index regardless of value in predicate

© 2007 Paul Baumgartel

With a histogram: example

- The optimizer can see that one value is extremely popular
- If this value is used in the predicate, a full table scan will occur
- If any other value is used, an index lookup will occur

© 2007 Paul Baumgartel

Histograms on non-indexed columns

- Despite emphasis on using histograms for indexed columns, they can also be useful for non-indexed columns
- Optimizer uses them in this case to help determine join order
- A row source that contains 400k rows is much less likely to appear early in a join order than one containing two rows

© 2007 Paul Baumgartel

Histograms can be great, but...

- They're not useful when bind variables are employed in code. Why?
  - They require knowledge of the actual value in the predicate, but binds don't provide it
  - Oracle does "bind variable peeking": at parse time, it inspects the value bound to the variable and feeds that to the optimizer *for all executions of the SQL*

© 2007 Paul Baumgartel

Cases for which histograms might be contraindicated

- On tables for which most SQL uses binds
- If you have reduced hard parsing by setting CURSOR_SHARING to FORCE or SIMILAR
- We all know that binds are preferable to literals, so in cases where histograms might be helpful you have to take great care and test, test, test

© 2007 Paul Baumgartel

Identifying histogram candidate columns

- First and foremost, it does no good, and increases overhead, to create histograms on columns whose data distribution is not skewed
- So don’t create histograms indiscriminately!

© 2007 Paul Baumgartel

Identifying histogram candidate columns, continued

- DBMS_STATS provides a “SKEWONLY” option
- It is supposed to create histograms only on columns with skewed data distributions
- Unfortunately, it doesn’t work very well
- Much better to determine data skew directly to your own satisfaction

© 2007 Paul Baumgartel

Identifying histogram candidate columns, continued

- There are various way to do this
- My preference is to find a query that someone else has written and use that
- Who better to steal from than Tom Kyte?
- The query on asktom.oracle.com produces a nice graphic representation of skew

© 2007 Paul Baumgartel

Identifying histogram candidate columns, continued

```
select wb, cnt,
  to_char(round( 100*cnt/(max(cnt) over ()),2),
  '999.00') rat,
  rpad( '*', 40*cnt/(max(cnt) over ()), '*' ) hist
  from (select wb,  count(*) cnt
  from (select width_bucket( r, 0,
  (select   count(distinct &cname)
  from &tname)+1,255) wb
  from (select dense_rank() over (order by &cname) r
  from &tname))
group by wb) order by wb
/
```

Substitution variables tname and cname are table and column names

© 2007 Paul Baumgartel

Candidate query results

WB	CNT	RAT	HIST
1	3033	.01	
2	49	.00	
3	23	.00	
4	27	.00	
5	515	.00	
6	684415	1.37	
7	15270	.03	
8	15757	.03	
9	43919	.09	
10	419413	.84	
11	9512	.02	
12	9479	.02	
13	4498	.01	
14	166645	.33	
15	2809	.01	
16	19315	.04	
17	9786	.02	
18	65466	.13	
19	1775111	3.55	*
20	598581	1.20	
21	31389	.06	
22	41223543	82.51	**********
23	37406226	74.87	**********
24	57905	.12	
25	14030	.03	

© 2007 Paul Baumgartel

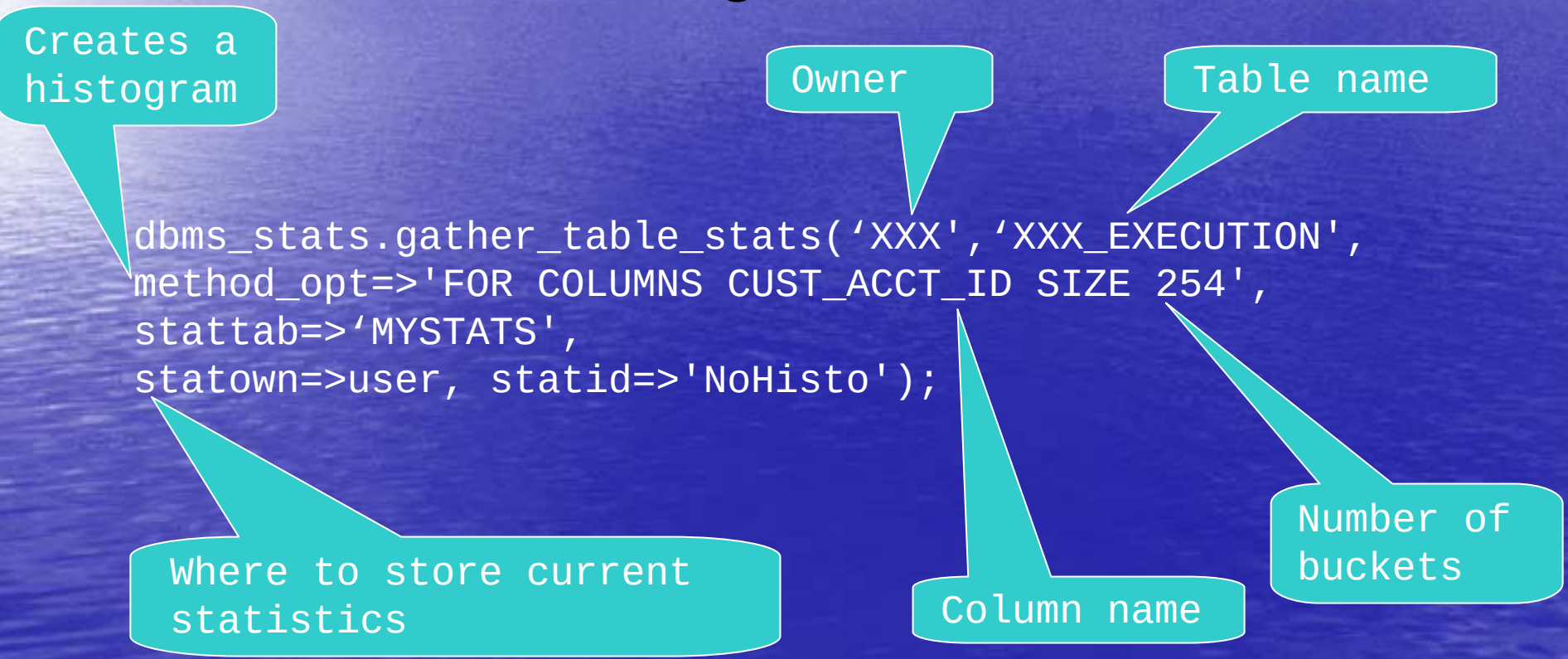
Candidate query results

- The previous slide shows that a couple of values are quite popular, making the column a good candidate for a histogram
- Remember to test!

© 2007 Paul Baumgartel

Creating histograms

- Generate for a single column



© 2007 Paul Baumgartel

Buckets and histogram types

- A histogram can have from 1 to 254 buckets
- The default level of column statistics is equivalent to a histogram with 1 bucket
- If the number of distinct values is <= 254, Oracle creates a *frequency histogram*
- Each “bucket” consists of a column value and the number of times it occurs

© 2007 Paul Baumgartel

Buckets and histogram types continued

- If number of distinct values > 254, Oracle creates a *height-balanced histogram*
- Each bucket represents ceil(row_count/num_buckets) rows, and the high value (end point) for that bucket is stored
- Popular values are identified by the fact that they appear as end points in more than one bucket
- Unpopular values can be completely obscured

© 2007 Paul Baumgartel

Loading and unloading statistics

- For testing, important to be able to switch between using and not using histograms
- Histograms are expensive to calculate, so we save them in a user stats table

```
dbms_stats.export_column_stats(‘XXX',‘XXX_EXECUTION',-
colname=>'CUST_ACCT_ID', stattab=>‘MYSTATS',-
statid=>'Histo', statown=>user)
```

© 2007 Paul Baumgartel

Loading and unloading statistics

- To prepare for test using alternate set of statistics:

```
dbms_stats.delete_column_stats(‘XXX',‘XXX_EXECUTION',
colname=>'CUST_ACCT_ID')
```

```
dbms_stats.import_column_stats(‘XXX',‘XXX_EXECUTION',
colname=>'CUST_ACCT_ID', stattab=>‘MYSTATS',
statid=>'NoHisto',statown=>user)
```

© 2007 Paul Baumgartel

Examining optimizer decisions

- Create a trace file showing optimizer plan consideration

```
alter session set events
'10053 trace name context forever, level 1';
```

© 2007 Paul Baumgartel