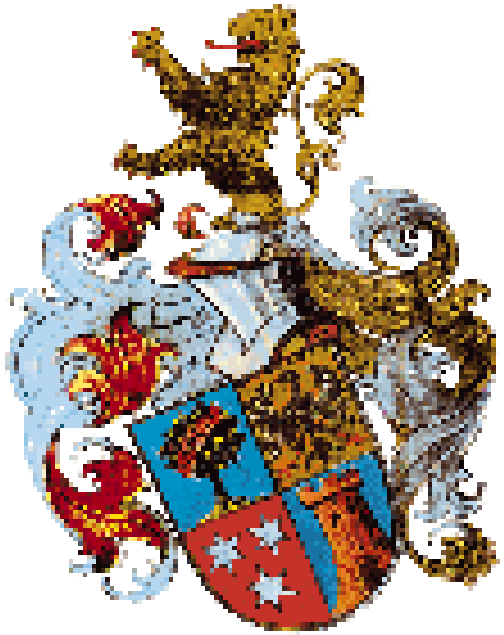
Active Statistics



Wolfgang Breitling
www.centrexcc.com

Who am I

Independent consultant since 1996
- specializing in Oracle and Peoplesoft setup, administration, and performance tuning

Member of the Oaktable Network

25+ years in database management
- DL/1, IMS, ADABAS, SQL/DS, DB2, Oracle

Oracle since 1993 (7.0.12)

OCP certified DBA - 7, 8, 8*i*, 9*i*

Mathematics major from University of Stuttgart

Focus

- Gather Statistics
  - Gather Partition Statistics
  - Gather Histograms
- Clone Statistics
- Modify or RYO Statistics
  - prepare_column_stats

Gather_Stats and the Stattab Table

- gather_object_stats(..., stattab=> , statid=>)
  the current statistics in the dictionary get saved into stattab under the given statid
  new statistics are gathered into the dictionary
- gather_system_stats(..., stattab=> , statid=>)
  the current statistics in the dictionary, if any, are unaffected
  new statistics are gathered directly into stattab under the given statid

Gather_System_Stats Strategy

1. **Gather system stats into a stattab table using fixed times and intervals with distinct statids e.g. schedule hourly or at certain times each day:**

   ```
   gather_system_stats(interval=>60,
   statid=>'S'||to_char(sysdate,'yyyymmddhh24miss'), … );
   ```

2. **Process this data, e.g. using averages, and use those values to set system statistics with set_system_stats. Make sure mreadtm is > sreadtm!**

   ```
   select statid, n3 cpuspeed, n1 sreadtim, n2 mreadtim, n11 mbrc
   from stats_table where type = 'S' and c4 = 'CPU_SERIO' order by statid
   ```

dbms_stats defaults

select sname, sval1, spare4 from sys.optstat_hist_control$

SNAME	SVAL1	SPARE4
SKIP_TIME		
TRACE		0
DEBUG		0
SYS_FLAGS		0
STATS_RETENTION	31	
CASCADE		DBMS_STATS.AUTO_CASCADE
ESTIMATE_PERCENT		DBMS_STATS.AUTO_SAMPLE_SIZE
DEGREE		NULL
METHOD_OPT		FOR ALL COLUMNS SIZE AUTO
NO_INVALIDATE		DBMS_STATS.AUTO_INVALIDATE
GRANULARITY		AUTO
AUTOSTATS_TARGET		AUTO
APPROXIMATE_NDV		TRUE
PUBLISH		TRUE
STALE_PERCENT		10
INCREMENTAL		FALSE
INCREMENTA_INTERNAL_CONTROL		TRUE

Words of Caution*

- ESTIMATE_PERCENT had a default of 100% on 9i. 10g defaults to DBMS_STATS.AUTO_SAMPLE_SIZE for sample size, […]. Small sample sizes are known to produce poor number of distinct values NDV on columns with skewed data (which are common), thus generate sub-optimal plans. Use an estimate of 100% if your window maintenance can afford it, even if that means gather statistics less often.
- METHOD_OPT had a default of "FOR ALL COLUMNS SIZE 1" on 9i, which basically meant NO HISTOGRAMS. 10g defaults to AUTO, which means DBMS_STATS decides in which columns a histogram may help to produce a better plan. It is known that in some cases, the effect of a histogram is adverse to the generation of a better plan.

* Metalink Note 465787.1. Emphasis all mine

Gather Statistics Performance (10.2.0.3)

gather_table_stats(…,method_opt=>'for all columns size 1', cascade=>true)

NAME	100	10	1	0.1	0.01	auto
sql execute elapsed time	22:55.4	01:53.9	02:14.5	01:36.5	02:17.1	04:42.0
DB CPU	06:26.0	01:22.9	00:58.3	00:47.1	00:59.1	02:40.3
session logical reads	475,464	384,039	279,303	52,801	65,639	719,550
index fast full scans (full)	0	4	4	4	4	6
table fetch by rowid	161	259	259	259	259	259
table scans (long tables)	1	1	1	1	2	3
sorts (rows)	72,002,374	7,208,584	2,320,365	1,934,490	1,948,871	3,598,961
physical reads direct temp tblsp	139,972	0	0	0	0	0
physical writes	139,972	0	0	0	0	0
workarea executions - onepass	5	0	0	0	0	0
workarea executions - optimal	14	32	31	31	32	33

Gather Statistics Performance (11.1.0.6)

gather_table_stats(…,method_opt=>'for all columns size 1', cascade=>true)

NAME	100	10	1	0.1	0.01	auto
sql execute elapsed time	21:57.5	02:05.1	01:42.1	01:57.1	02:06.4	02:00.8
DB CPU	06:40.7	01:07.6	00:40.9	00:42.8	00:43.9	01:00.3
session logical reads	489,324	397,822	293,683	66,384	88,169	395,225
index fast full scans (full)	0	4	4	4	4	4
table fetch by rowid	155	215	215	215	215	313
table scans (long tables)	1	1	1	1	2	1
sorts (rows)	72,000,017	7,204,143	2,258,623	1,982,680	1,965,031	1,855,556
physical reads direct temp tblsp	139,938	0	0	0	0	0
physical writes	139,938	0	0	0	0	0
workarea executions - onepass	5	0	0	0	0	0
workarea executions - optimal	10	28	27	27	28	50

Gather Statistics Performance (10.2.0.3)

gather_table_stats(…,method_opt=>'for all columns size 1', cascade=>false)

NAME	100	10	1	0.1	0.01	auto
sql execute elapsed time	19:43.6	02:18.6	01:20.5	01:14.6	01:57.9	04:15.3
DB CPU	03:34.1	01:18.2	00:48.1	00:35.8	00:51.3	02:34.2
session logical reads	372,870	372,817	273,456	46,930	59,223	716,466
index fast full scans (full)	0	0	0	0	0	2
table fetch by rowid	42	42	42	42	42	42
table scans (long tables)	1	1	1	1	2	3
sorts (rows)	44,002,358	4,406,505	437,617	45,654	57,842	1,660,336
physical reads direct temp tblsp	100,551	0	0	0	0	0
physical writes	100,551	0	0	0	0	0
workarea executions - onepass	1	0	0	0	0	0
workarea executions - optimal	6	7	7	7	8	9

Gather Statistics Performance (11.1.0.6)

gather_table_stats(…,method_opt=>'for all columns size 1', cascade=>false)

NAME	100	10	1	0.1	0.01	auto
sql execute elapsed time	18:53.9	01:28.8	01:31.4	02:38.2	01:49.9	01:31.3
DB CPU	03:55.2	00:53.3	00:49.4	00:32.1	00:34.2	00:50.1
session logical reads	386,797	386,638	288,086	60,704	84,450	389,635
index fast full scans (full)	0	0	0	0	0	0
table fetch by rowid	60	50	50	52	50	67
table scans (long tables)	1	1	1	1	2	1
sorts (rows)	44,000,000	4,390,067	439,043	42,820	60,929	54
physical reads direct temp tblsp	100,512	0	0	0	0	0
physical writes	100,512	0	0	0	0	0
workarea executions - onepass	1	0	0	0	0	0
workarea executions - optimal	2	3	3	6	4	6

Gather Statistics

- An estimate_percent < 100 does perform faster than a full compute.
- The assertion that essentially all blocks are read as long as estimate_percent > 100/average rows per block is correct though.
- The difference comes from the sort(s) required to get the column stats, whether collecting histograms (for all columns size auto) or not (for all columns size 1).
- That also means that, contrary to polular belief, estimate_percent > 50 is not equivalent to a full compute.

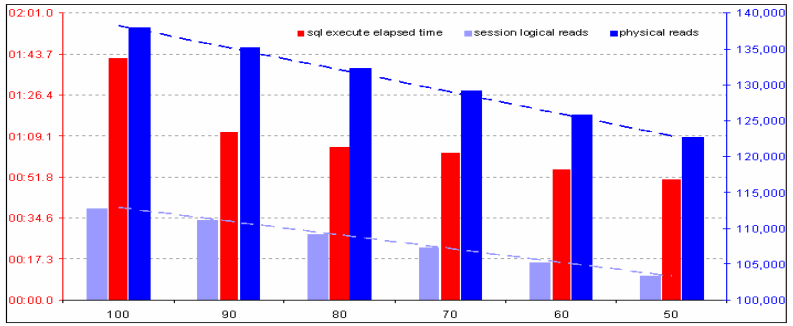


Table Statistics Accuracy (10.2.0.3)

gather_table_stats(…,method_opt=>'for all columns size 1', cascade=>true)

est%	sample	num_rows	blocks	rowlen	error*
100	4,000,000	4,000,000	375,980	232	
10	399,687	3,996,870	375,980	231	0.1%
1	39,960	3,996,000	375,980	232	0.1%
0.1	4,016	4,016,000	375,980	231	0.4%
0.01	4,786	3,991,524	375,980	232	0.2%
auto	451,918	3,996,024	375,980	231	0.1%

*** error = abs(actual-estimate)/actual**

Table Statistics Accuracy (11.1.0.6)

gather_table_stats(…,method_opt=>'for all columns size 1', cascade=>true)

est%	sample	num_rows	blocks	rowlen	error*
100	4,000,000	4,000,000	375,980	232	
10	398,677	3,986,770	375,980	231	0.3%
1	40,349	4,034,900	375,980	231	0.9%
0.1	4,001	4,001,000	375,980	232	0.0%
0.01	4,891	3,951,928	375,980	231	1.2%
auto	4,000,000	4,000,000	375,980	232	0.0%

*** error = abs(actual-estimate)/actual**

Column Distributions

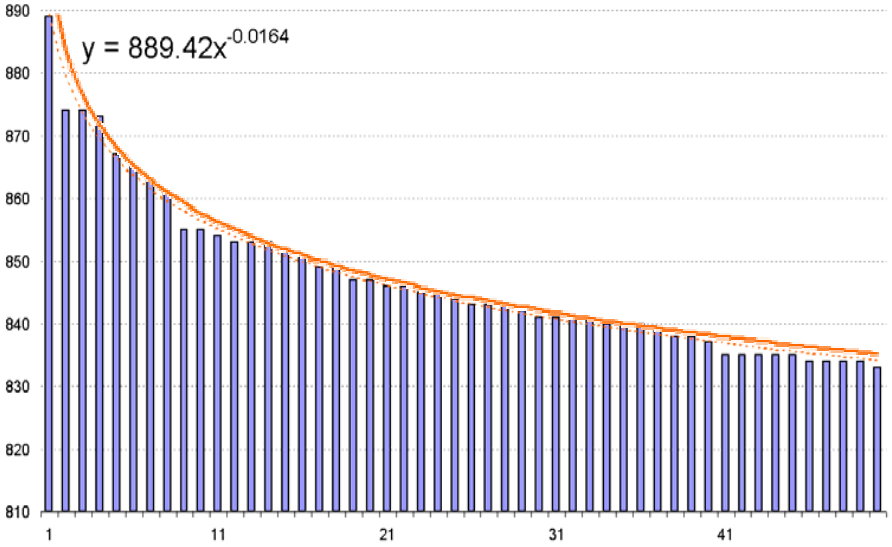
Name	Null?	Type
C1	NOT NULL	VARCHAR2(8)
C2	NOT NULL	VARCHAR2(8)
C3		VARCHAR2(16)
C4		VARCHAR2(30)
C5		VARCHAR2(8)
C6		VARCHAR2(30)
FY		NUMBER
AP		NUMBER
D1		DATE
FILLER		VARCHAR2(200)

index	column
T2_U	C1
	C2
	C3
	C4
	D1
	FY
	AP

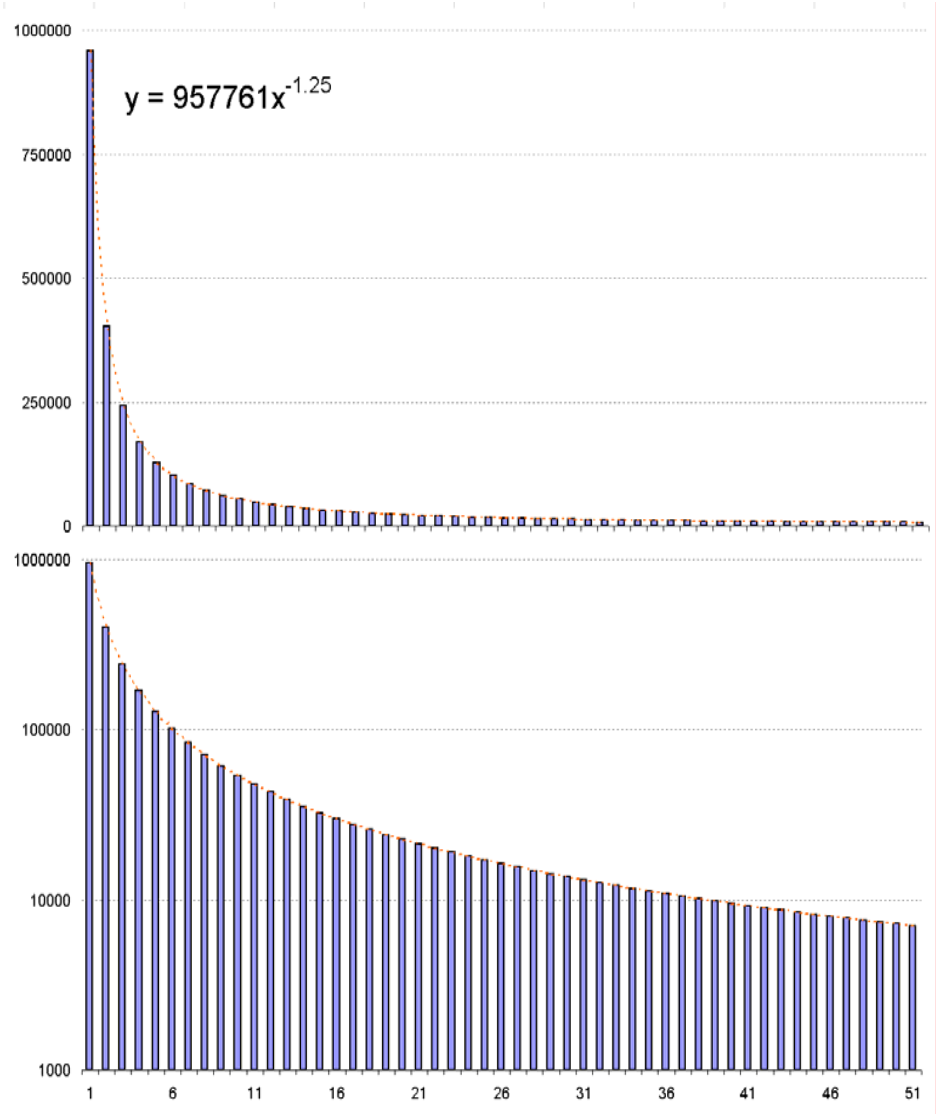
index	column
T2_I1	FY
	AP
	C1
T2_I2	FY
	AP
	C3
T2_I3	FY
	AP
	C5

Column Distributions

Normal Distribution



Log-Normal Distribution



Column Statistics Accuracy

random-"uniform" distribution

10.2.0.3

est%	sample	NDV	low	high	NDV	low	high
100	4,000,000	507,993	00314515	01341929			
10	399,687	353,053	00314515	01252878	30.5%	0.0%	6.6%
1	39,960	325,875	00380916	01231540	35.9%	21.1%	8.2%
0.1	4,016	236,012	00426333	01145732	53.5%	35.6%	14.6%
0.01	4,786	309,390	00361058	01127320	39.1%	14.8%	16.0%
auto	451,918	357,099	00440915	01202777	29.7%	40.2%	10.4%

Column Statistics Accuracy

random-"uniform" distribution

11.1.0.6

est%	sample	NDV	low	high	NDV	low	high
100	4,000,000	507,993	00314515	01341929			
10	398,677	352,949	00348801	01226439	30.5%	10.9%	8.6%
1	40,349	329,932	00397784	01328241	35.1%	26.5%	1.0%
0.1	4,001	346,985	00356860	01175345	31.7%	13.5%	12.4%
0.01	4,891	297,543	00398559	01139450	41.4%	26.7%	15.1%
auto	4,000,000	507,424	00314515	01341929	0.1%	0.0%	0.0%

See also Greg Rahn's and the Optimizer Development Team's blogs

Column Statistics Accuracy

normal distribution

10.2.0.3

est%	sample	NDV	low	high	NDV	low	high
100	4,000,000	15,156	004379	025303			
10	399,687	12,464	005288	024977	17.8%	20.8%	1.3%
1	39,960	9,119	006657	023001	39.8%	52.0%	9.1%
0.1	4,016	7,572	007231	022025	50.0%	65.1%	13.0%
0.01	4,786	7,238	008090	021944	52.2%	84.7%	13.3%
auto	45,229	9,286	008187	023403	38.7%	87.0%	7.5%

Column Statistics Accuracy

normal distribution

11.1.0.6

est%	sample	NDV	low	high	NDV	low	high
100	4,000,000	15,156	004379	025303			
10	398,677	12,497	004379	024726	17.5%	0.0%	2.3%
1	40,349	9,075	006892	022930	40.1%	57.4%	9.4%
0.1	4,001	7,203	008073	022747	52.5%	84.4%	10.1%
0.01	4,891	7,569	008104	021284	50.1%	85.1%	15.9%
auto	4,000,000	15,156	004379	025303	0.0%	0.0%	0.0%

Column Statistics Accuracy

lognormal distribution

10.2.0.3

est%	sample	NDV	low	high	NDV	low	high
100	4,000,000	7,994	00002527	09997860			
10	399,687	7,365	00002527	09997860	7.9%	0.0%	0.0%
1	39,960	3,033	00003581	09997860	62.1%	41.7%	0.0%
0.1	4,016	718	00030383	09997860	91.0%	1102.3%	0.0%
0.01	4,786	792	00003581	09995231	90.1%	41.7%	0.0%
auto	4,491	770	00033746	09992981	90.4%	1235.4%	0.0%

Column Statistics Accuracy

lognormal distribution

11.1.0.6

est%	sample	NDV	low	high	NDV	low	high
100	4,000,000	7,994	00002527	09997860			
10	398,677	7,401	00002527	09997860	7.4%	0.0%	0.0%
1	40,349	3,097	00003581	09997860	61.3%	41.7%	0.0%
0.1	4,001	730	00002527	09997860	90.9%	0.0%	0.0%
0.01	4,891	843	00016130	09995231	89.5%	538.3%	0.0%
auto	4,000,000	7,994	00002527	09997860	0.0%	0.0%	0.0%

Gathering Histograms

What, if any, is the difference between the following ways of gathering histograms on multiple columns?

❶ gather_table_stats(…, method_opt=>'for columns c1 size 254');
gather_table_stats(…, method_opt=>'for columns c2 size 254');
gather_table_stats(…, method_opt=>'for columns c3 size 254');
gather_table_stats(…, method_opt=>'for columns c4 size 254');

❷ gather_table_stats(…, method_opt=>'for columns c1,c2,c3,c4 size 254');

❸ gather_table_stats(…, method_opt=>'for columns size 254 c1,c2,c3,c4');

Gathering Histograms

NAME	❶ 100	❷ 100	❸ 100	❶ auto	❷ auto	❸ auto
sql execute elapsed time	07:48.1	03:16.0	03:25.7	06:07.5	01:37.3	01:42.1
DB CPU	05:20.0	02:26.2	02:29.9	02:43.0	00:47.5	00:48.0
consistent gets	1,575,736	457,469	458,000	430,163	142,869	140,752
db block gets	2,416	1,314	2,161	10,141	5,350	6,493
session logical reads	1,578,152	458,783	460,161	440,304	148,219	147,245
physical reads	1,721,051	596,635	596,642	1,479,241	407,921	406,051
table fetch by rowid	177	42	42	632	58	58
table scans (long tables)	4	1	1	4	1	1
table scans (short tables)	104	26	26	142	43	43
workarea executions - onepass	6	2	2	8	8	8
workarea executions - optimal	24	28	28	91	49	49

Gathering Histograms

		❶			❷			❸		
	col	NDV	bkts		NDV	bkts		NDV	bkts	
100	C1	7,994	254	H	7,994	75	H	7,994	254	H
	C2	3,999	254	H	3,999	75	H	3,999	254	H
	C3	507,993	254	H	507,993	75	H	507,993	254	H
	C4	399,986	254	H	399,986	254	H	399,986	254	H
auto	C1	1,050	254	H	844	75	H	1,035	254	H
	C2	206	205	F	174	75	H	196	195	F
	C3	366,478	254	H	362,236	75	H	359,990	254	H
	C4	377,364	254	H	377,847	254	H	377,153	254	H

Gather Partition Statistics

```
gather_table_stats(user, 'P1', cascade=>true,
method_opt=>'for all columns size 1',  estimate_percent=>100);
```

Elapsed: 00:23:14.37

table_name	rows	blks	empty	avrow	last_analyzed
P1	4,000,000	129,736	0	224	2008-01-27 12:16:36
P1 P1_2006_Q1	500,731	16,217	0	225	2008-01-27 11:54:05
P1 P1_2006_Q2	498,944	16,217	0	224	2008-01-27 11:54:18
P1 P1_2006_Q3	499,734	16,217	0	224	2008-01-27 11:54:29
P1 P1_2006_Q4	500,574	16,217	0	224	2008-01-27 11:54:43
P1 P1_2007_Q1	500,323	16,217	0	224	2008-01-27 11:54:54
P1 P1_2007_Q2	501,174	16,217	0	225	2008-01-27 11:55:05
P1 P1_2007_Q3	498,815	16,217	0	224	2008-01-27 11:55:16
P1 P1_2007_Q4	499,705	16,217	0	225	2008-01-27 11:55:27

Gather Partition Statistics

```
gather_table_stats(user, 'P1', 'P1_2007_Q4', cascade=>true,
method_opt=>'for all columns size 1',  estimate_percent=>100);
```

Elapsed: 00:20:21.90

table_name	rows	blks	empty	avrow	last_analyzed
P1	4,000,000	129,736	0	224	2008-01-27 13:40:24
P1 P1_2006_Q1	500,731	16,217	0	225	2008-01-27 11:54:05
P1 P1_2006_Q2	498,944	16,217	0	224	2008-01-27 11:54:18
P1 P1_2006_Q3	499,734	16,217	0	224	2008-01-27 11:54:29
P1 P1_2006_Q4	500,574	16,217	0	224	2008-01-27 11:54:43
P1 P1_2007_Q1	500,323	16,217	0	224	2008-01-27 11:54:54
P1 P1_2007_Q2	501,174	16,217	0	225	2008-01-27 11:55:05
P1 P1_2007_Q3	498,815	16,217	0	224	2008-01-27 11:55:16
P1 P1_2007_Q4	499,705	16,217	0	225	2008-01-27 13:20:25

Gather Partition Statistics

```
gather_table_stats(user, 'P1', 'P1_2007_Q4', cascade=>true,
method_opt=>'for all columns size 1',  estimate_percent=>100,
granularity=>'partition');
```

Elapsed: 00:00:25.56

table_name	rows	blks	empty	avrow	last_analyzed
P1	4,000,000	129,736	0	224	2008-01-27 13:40:24
P1 P1_2006_Q1	500,731	16,217	0	225	2008-01-27 11:54:05
P1 P1_2006_Q2	498,944	16,217	0	224	2008-01-27 11:54:18
P1 P1_2006_Q3	499,734	16,217	0	224	2008-01-27 11:54:29
P1 P1_2006_Q4	500,574	16,217	0	224	2008-01-27 11:54:43
P1 P1_2007_Q1	500,323	16,217	0	224	2008-01-27 11:54:54
P1 P1_2007_Q2	501,174	16,217	0	225	2008-01-27 11:55:05
P1 P1_2007_Q3	498,815	16,217	0	224	2008-01-27 11:55:16
P1 P1_2007_Q4	499,705	16,217	0	225	2008-01-27 13:48:13

The CBO and Partition Pruning

```
select c5, c6, fy, ap, to_char(d1,'yyyy-mm-dd') d1
from p1 where fy = 2007 and ap = 10
and c1 = '00873' and c2 = '013912'
and c3 = '000000740444' and c4 = '000000974796'
```

```
ELAPSED    CPU  exec  LIO  PIO  rows SQL_TEXT
-------- ------ ----- ---- ---- ----- ---------------------------------
   0.036  0.013     1   14    2     1 select c5, c6, fy, ap, to_char(d1,'yy


| Id  | Operation                          | Name | Rows  | Pstart| Pstop |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |      |       |       |       |
|   1 |  PARTITION RANGE SINGLE            |      |     1 |     8 |     8 |
|   2 |   TABLE ACCESS BY LOCAL INDEX ROWID| P1   |     1 |     8 |     8 |
|*  3 |    INDEX RANGE SCAN                | P1_U |     1 |     8 |     8 |
-------------------------------------------------------------------------

   3 - access("C1"='00873' AND "C2"='013912' AND "C3"='000000740444' AND
              "C4"='000000974796' AND "FY"=2007 AND "AP"=10)
       filter(("AP"=10 AND "FY"=2007))
```

Partition Statistics – Add a Partition

```
alter table p1 add partition "P1_2008_Q1"
    values less than (2008,4)

gather_schema_stats(user,options=>'GATHER AUTO');
```

table_name	rows	blks	empty	avrow	last_analyzed
P1	4,001,376	131,264	0	225	2007-12-05 14:50:41
P1 P1_2006_Q1	500,731	16,217	0	225	2007-12-01 17:53:36
P1 P1_2006_Q2	498,944	16,217	0	224	2007-12-01 17:53:57
P1 P1_2006_Q3	499,734	16,981	0	224	2007-12-01 17:54:18
P1 P1_2006_Q4	500,574	16,217	0	224	2007-12-01 17:54:40
P1 P1_2007_Q1	500,323	16,217	0	224	2007-12-01 17:55:01
P1 P1_2007_Q2	501,174	16,217	0	225	2007-12-01 17:55:26
P1 P1_2007_Q3	498,815	16,089	0	224	2007-12-01 17:55:48
P1 P1_2007_Q4	499,705	17,109	0	225	2007-12-01 17:56:11
P1 P1_2008_Q1	0	0	0	0	2007-12-05 14:46:12

The CBO and Partition Pruning

```
select c5, c6, fy, ap, to_char(d1,'yyyy-mm-dd') d1
from p1 where fy = 2008 and ap = 1
and c1 = '00873‘ and c2 = '013912‘
and c3 = '000000740444‘ and c4 = '000000974796'
```

```
 ELAPSED    CPU  exec    LIO     PIO  rows SQL_TEXT
------- ------ ----- ------ ------- ----- --------------------------------
   1.058  0.918     1  2,561   1,713     1 select fy, ap, c6, c5, to_char(d1,


| Id  | Operation                          | Name  | Rows  | Pstart| Pstop |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |       |       |       |       |
|   1 |  PARTITION RANGE SINGLE            |       |     1 |     9 |     9 |
|*  2 |   TABLE ACCESS BY LOCAL INDEX ROWID| P1    |     1 |     9 |     9 |
|*  3 |    INDEX RANGE SCAN                | P1_I1 |     1 |     9 |     9 |
-------------------------------------------------------------------------

   2 - filter(("C2"='016374' AND "C3"='000000809624' AND "C4"='000000719086'))
   3 - access("FY"=2008 AND "AP"=1 AND "C1"='00797')
       filter("C1"='00797')
```

Cloning Statistics

```
dbms_stats.export_table_stats(user,'P1','P1_2007_Q4',
    statid=>'P1_2007_Q4',cascade=>true,stattab=>'stats_table');
dbms_stats.import_table_stats(user,'P1','P1_2008_Q1',
    statid=>'P1_2007_Q4',cascade=>true,stattab=>'stats_table');
```

table_name	rows	blks	empty	avrow	last_analyzed
P1	4,001,376	131,264	0	225	2007-12-05 14:50:41
P1 P1_2006_Q1	500,731	16,217	0	225	2007-12-01 17:53:36
P1 P1_2006_Q2	498,944	16,217	0	224	2007-12-01 17:53:57
P1 P1_2006_Q3	499,734	16,981	0	224	2007-12-01 17:54:18
P1 P1_2006_Q4	500,574	16,217	0	224	2007-12-01 17:54:40
P1 P1_2007_Q1	500,323	16,217	0	224	2007-12-01 17:55:01
P1 P1_2007_Q2	501,174	16,217	0	225	2007-12-01 17:55:26
P1 P1_2007_Q3	498,815	16,089	0	224	2007-12-01 17:55:48
P1 P1_2007_Q4	499,705	17,109	0	225	2007-12-01 17:56:11
P1 P1_2008_Q1	0	0	0	0	2007-12-05 14:46:12

Cloning Statistics

```
Update stats_table set c2 = 'P1_2008_Q1'
      where statid = 'P1_2007_Q4' and c2 = 'P1_2007_Q4';
commit;
dbms_stats.import_table_stats(user,'P1','P1_2008_Q1',
      statid=>'P1_2007_Q4',cascade=>true,stattab=>'stats_table');
```

table_name	rows	blks	empty	avrow	last_analyzed
P1	4,001,376	131,264	0	225	2007-12-05 14:50:41
P1 P1_2006_Q1	500,731	16,217	0	225	2007-12-01 17:53:36
P1 P1_2006_Q2	498,944	16,217	0	224	2007-12-01 17:53:57
P1 P1_2006_Q3	499,734	16,981	0	224	2007-12-01 17:54:18
P1 P1_2006_Q4	500,574	16,217	0	224	2007-12-01 17:54:40
P1 P1_2007_Q1	500,323	16,217	0	224	2007-12-01 17:55:01
P1 P1_2007_Q2	501,174	16,217	0	225	2007-12-01 17:55:26
P1 P1_2007_Q3	498,815	16,089	0	224	2007-12-01 17:55:48
P1 P1_2007_Q4	499,705	17,109	0	225	2007-12-01 17:56:11
P1 P1_2008_Q1	499,705	17,109	0	225	2007-12-01 17:56:11

Cloning Statistics – Method "exp"

exp scott/tiger,file=clone,rows=no,tables=P1:P1_2007_Q4
. . exporting table P1
EXP-00091: Exporting questionable statistics.

clone.dmp:

ANALSTATS TR "P1"
U BEGIN
DBMS_STATS.SET_TABLE_STATS(NULL,'"P1"','"P1_2007_Q4"',NULL,NULL,
0,0,0,2); END;
ANALSTATS TR "P1"
ýÿw DECLARE SREC DBMS_STATS.STATREC; BEGIN SREC.MINVAL := NULL;
SREC.MAXVAL := NULL; SREC.EAVS := 0; SREC.CHVALS := NULL; #
SREC.NOVALS := DBMS_STATS.NUMARRAY(0,0&); SREC.BKVALS :=
DBMS_STATS.NUMARRAY(0,1o); SREC.EPC := 2;
DBMS_STATS.SET_COLUMN_STATS(NULL,'"P1"','"C1"','"P1_2007_Q4"',
NULL,NULL,0,0,0,srec,0,2); END;
...

Cloned Statistics

```
select c5, c6, fy, ap, to_char(d1,'yyyy-mm-dd') d1
from p1 where fy = 2008 and ap = 1
and c1 = '00873‘ and c2 = '013912‘
and c3 = '000000740444‘ and c4 = '000000974796'
```

```
 ELAPSED    CPU  exec    LIO     PIO  rows SQL_TEXT
-------- ------ ----- ------ ------- ----- --------------------------------
   0.968  0.968     1  2,564   1,724     1 select fy, ap, c6, c5, to_char(d1,


| Id  | Operation                          | Name  | Rows  | Pstart| Pstop |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |       |       |       |       |
|   1 |  PARTITION RANGE SINGLE            |       |     1 |     9 |     9 |
|*  2 |   TABLE ACCESS BY LOCAL INDEX ROWID| P1    |     1 |     9 |     9 |
|*  3 |    INDEX RANGE SCAN                | P1_I1 |     1 |     9 |     9 |
-------------------------------------------------------------------------

   2 - filter(("C3"='000000809624' AND "C4"='000000719086' AND "C2"='016374'))
   3 - access("FY"=2008 AND "AP"=1 AND "C1"='00797')
       filter("C1"='00797')
```

Cloned Statistics

BASE STATISTICAL INFORMATION

****

Table Stats::

Table: P1 Alias: P1 Partition [8]

#Rows: 499705 #Blks: 17109 AvgRowLen: 225.00

#Rows: 499705 #Blks: 17109 AvgRowLen: 225.00

Index Stats::

Index: P1_I1 Col#: 7 8 9 1 PARTITION [8]

LVLS: 2 #LB: 2231 #DK: 296545 LB/K: 1.00 DB/K: 1.00 CLUF: 499667.00

LVLS: 2 #LB: 2231 #DK: 296545 LB/K: 1.00 DB/K: 1.00 CLUF: 499667.00

Index: P1_U Col#: 1 2 3 4 9 7 8 PARTITION [8]

LVLS: 2 #LB: 4462 #DK: 499705 LB/K: 1.00 DB/K: 1.00 CLUF: 499675.00

LVLS: 2 #LB: 4462 #DK: 499705 LB/K: 1.00 DB/K: 1.00 CLUF: 499675.00

Cloned Statistics

```
Column (#8): AP(NUMBER)
  AvgLen: 3.00 NDV: 3 Nulls: 0 Density: 0.33333 Min: 10 Max: 12
 Using prorated density: 1.0006e-06 of col #8 as selectivity of out-of-range value
     pred
[…]
 Access Path: index (skip-scan)
   SS sel: 2.4629e-15  ANDV (#skips): 530
   SS io: 530.00 vs. index scan io: 1.00
   Skip Scan rejected
 Access Path: index (RangeScan)
   Index: P1_I1
   resc_io: 4.00  resc_cpu: 26374
   ix_sel: 2.0024e-12  ix_sel_with_filters: 2.4629e-15
   Cost: 4.04  Resp: 4.04  Degree: 1
```

Cloned Statistics

```
  Access Path: index (skip-scan)
    SS sel: 2.4833e-30  ANDV (#skips): 530
    SS io: 530.00 vs. index scan io: 1.00
    Skip Scan rejected
  Access Path: index (RangeScan)
    Index: P1_U
    resc_io: 4.00  resc_cpu: 26374
    ix_sel: 1.2402e-18  ix_sel_with_filters: 2.4833e-30
    Cost: 4.04  Resp: 4.04  Degree: 1
[…]
  Best:: AccessPath: IndexRange  Index: P1_I1
         Cost: 4.04  Degree: 1  Resp: 4.04  Card: 0.00  Bytes: 0
```

Prepare and set Column Stats

```
DECLARE
 SREC DBMS_STATS.STATREC;
 NOVALS DBMS_STATS.NUMARRAY;
BEGIN
 SREC.EAVS := 0;
 SREC.CHVALS := NULL;
 SREC.EPC := 2;
 NOVALS := DBMS_STATS.NUMARRAY(2008,2008);
 SREC.BKVALS := DBMS_STATS.NUMARRAY(0,1);
 DBMS_STATS.PREPARE_COLUMN_VALUES (SREC,NOVALS);
 DBMS_STATS.SET_COLUMN_STATS(user,'P1','FY','P1_2008_Q1',NULL,NULL,1,1,0,srec,4,6);

SREC.EAVS := 0;
 SREC.CHVALS := NULL;
 SREC.EPC := 2;
 NOVALS := DBMS_STATS.NUMARRAY(1,3);
 SREC.BKVALS := DBMS_STATS.NUMARRAY(0,1);
 DBMS_STATS.PREPARE_COLUMN_VALUES (SREC,NOVALS);
 DBMS_STATS.SET_COLUMN_STATS(user,'P1','AP','P1_2008_Q1',NULL,NULL,3,1/3,0,srec,3,6);
END;
```

Prepare and set Column Stats

table	partition	column	NDV	density	LOW	HIGH
P1	P1_2008_Q1	C1	813	1.2300E-03	00336	01255
		C2	12,770	7.8309E-05	005680	024231
[…]						
		FY	1	1.0000E+00	2008	2008
		AP	3	3.3333E-01	1	3

The CBO and Partition Pruning

```
select c5, c6, fy, ap, to_char(d1,'yyyy-mm-dd') d1
from p1 where fy = 2008 and ap = 1
and c1 = '00873‘ and c2 = '013912‘
and c3 = '000000740444‘ and c4 = '000000974796'
```

```
 ELAPSED    CPU  exec   LIO   PIO  rows SQL_TEXT
-------- ------ ----- ----- ----- ----- --------------------------------
   0.031  0.017     1     6     3     1 select fy, ap, c6, c5, to_char(d1,


| Id  | Operation                          | Name | Rows  | Pstart| Pstop |
-------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |      |       |       |       |
|   1 |  PARTITION RANGE SINGLE            |      |     1 |     9 |     9 |
|   2 |   TABLE ACCESS BY LOCAL INDEX ROWID| P1   |     1 |     9 |     9 |
|*  3 |    INDEX RANGE SCAN                | P1_U |     1 |     9 |     9 |
-------------------------------------------------------------------------

   3 - access("C1"='00797' AND "C2"='016374' AND "C3"='000000809624' AND
              "C4"='000000719086' AND "FY"=2008 AND "AP"=1)
       filter(("AP"=1 AND "FY"=2008))
```

More Prepare Column Stats

```
DECLARE
 SREC DBMS_STATS.STATREC;
 CHVALS DBMS_STATS.CHARARRAY;
 DTVALS DBMS_STATS.DATEARRAY;
BEGIN
 SREC.EAVS := 0;
 SREC.CHVALS := NULL;
 SREC.EPC := 2;
 CHVALS := DBMS_STATS.CHARARRAY('1234','5678');
 SREC.BKVALS := DBMS_STATS.NUMARRAY(0,1);
 DBMS_STATS.PREPARE_COLUMN_VALUES (SREC,CHVALS);
 DBMS_STATS.SET_COLUMN_STATS(…);

 SREC.EAVS := 0;
 SREC.CHVALS := NULL;
 SREC.EPC := 2;
 DTVALS := DBMS_STATS.DATEARRAY(date '2007-10-01',date '2007-12-31');
 SREC.BKVALS := DBMS_STATS.NUMARRAY(0,1);
 DBMS_STATS.PREPARE_COLUMN_VALUES (SREC,DTVALS);
 DBMS_STATS.SET_COLUMN_STATS(…);
END;
```

Create a Frequency Histogram

val	count
D	2852104
H	22
I	11414
N	5993
Z	28

```
DECLARE
 SREC DBMS_STATS.STATREC;
 CHVALS DBMS_STATS.CHARARRAY;
BEGIN
 SREC.EAVS := 0;
 SREC.CHVALS := NULL;
 SREC.EPC := 5;
 CHVALS := DBMS_STATS.CHARARRAY('D','H','I','N','Z');
 SREC.BKVALS := DBMS_STATS.NUMARRAY(2852104,22,11414,5993,28);
 DBMS_STATS.PREPARE_COLUMN_VALUES (SREC,CHVALS);
 DBMS_STATS.SET_COLUMN_STATS(…);
END;
```

Sampled Histograms

table	column	NDV	density	bkts	H	sample
T4	N1	29	4.9583E-06	29	F	4,798
	N2	99,978	1.0004E-05	254	H	4,798
	C1	95,057	1.0520E-05	1	N	4,798
	C2	95,042	1.0522E-05	1	N	4,798

tbl	col	EP	value	card
T4	N1	2325	0	2325
T4	N1	3066	1	741
T4	N1	3553	2	487
…				
T4	N1	4782	19	6
T4	N1	4784	20	2
T4	N1	4786	21	2
T4	N1	4789	22	3
T4	N1	4791	23	2
T4	N1	4794	24	3
T4	N1	4795	26	1
T4	N1	4796	27	1
T4	N1	4797	29	1
T4	N1	4798	32	1

```
SELECT L1.N1, L2.N1, SUM (A.N3) N3_SUM
FROM T3 A, T4 L1, T4 L2, T5 L
  WHERE A.N1 = 2005
    AND A.N2 BETWEEN 1 AND 12
    AND L1.N1 = 26
    AND A.C4 >= L1.C1
    AND A.C4 <= L1.C2
    AND ( L1.N2 BETWEEN 1240000000 AND 1300000000
       OR L1.N2 BETWEEN 1310000000 AND 1400000000
       OR L1.N2 BETWEEN 1410000000 AND 1420000000
       OR L1.N2 BETWEEN 1430000000 AND 1440000000 )
    AND L2.N1 = 23
    AND A.C3 >= L2.C1
    AND A.C3 <= L2.C2
    AND ( L2.N2 BETWEEN 1250000000 AND 1379999999
       OR L2.N2 BETWEEN 1400000000 AND 1550000000 )
    AND L.N1 = 2
    AND A.C1 = L.C1
    AND L.N2 BETWEEN 1750000000 AND 2000000000
    AND A.C6 = 'USD'
    AND A.C7 = ' '
GROUP BY L1.N1, L2.N1
```

Sampled Histograms

elapsed	cpu	LIO	PIO	rows
5,283.146	763.920	1,616,466	1,074,240	1

id	card	operation	starts	rows
0		SELECT STATEMENT ALL_ROWS		
1	1	SORT GROUP BY NOSORT	1	1
2	1	NESTED LOOPS	1	3,099
3	1	NESTED LOOPS	1	60,771
4	2	MERGE JOIN CARTESIAN	1	92
5AF	1	INDEX RANGE SCAN T4_U [L1]	1	4
6	24	BUFFER SORT	4	92
7AF	24	INDEX RANGE SCAN T4_U [L2]	1	23
8F	1	TABLE ACCESS BY INDEX ROWID T3 [A]	92	60,771
9AF	1	INDEX RANGE SCAN T3_F [A]	92	1,534,204
10AF	1	INDEX RANGE SCAN T5_U [L]	60,771	3,099

Pro-rate the Frequency H'gram

```
DECLARE
 srec DBMS_STATS.STATREC;
 l_distcnt NUMBER;
 l_density NUMBER;
 l_nullcnt NUMBER;
 l_avgclen NUMBER;
 l_numrows NUMBER;
 l_prorate number;
BEGIN
  DBMS_STATS.GET_COLUMN_STATS ( ownname => user, tabname => 'T4', colname => 'N1',
    distcnt => l_distcnt, density => l_density, nullcnt => l_nullcnt, avgclen => l_avgclen
    , srec => SREC );
  select num_rows into l_numrows from user_tables where table_name = 'T4';
  l_prorate := l_numrows/SREC.BKVALS(SREC.EPC);
  FOR I in 1..srec.epc LOOP
    SREC.BKVALS(I) := round(SREC.BKVALS(I)*l_prorate,0);
  END LOOP;
  DBMS_STATS.SET_COLUMN_STATS ( ownname => user, tabname => 'T4', colname => 'N1',
    distcnt => l_distcnt, density => l_density, nullcnt => l_nullcnt, avgclen => l_avgclen,
    srec => SREC, no_invalidate => FALSE, force => TRUE);
END;
```

Sampled Histograms

table	column	EP	value	card
T4	N1	48458	0	48458
T4	N1	63902	1	15444
T4	N1	74052	2	10150
…				
T4	N1	99667	19	126
T4	N1	99708	20	41
T4	N1	99750	21	42
T4	N1	99812	22	62
T4	N1	99854	23	42
T4	N1	99917	24	63
T4	N1	99937	26	20
T4	N1	99958	27	21
T4	N1	99979	29	21
T4	N1	100000	32	21

Sampled Histograms

elapsed	cpu	LIO	PIO	rows
1,316.790	296.169	789,222	268,777	1

id	card	operation	starts	ROWS
0		SELECT STATEMENT ALL_ROWS		
1	1	SORT GROUP BY NOSORT	1	1
2	1	NESTED LOOPS	1	3,099
3	1	NESTED LOOPS	1	18,522
4	1	NESTED LOOPS	1	363,409
5AF	24	INDEX RANGE SCAN T4_U [L2]	1	23
6F	1	TABLE ACCESS BY INDEX ROWID T3 [A]	23	363,409
7AF	1	INDEX RANGE SCAN T3_F [A]	23	383,551
8AF	1	INDEX RANGE SCAN T5_U [L]	363,409	18,522
9AF	1	INDEX RANGE SCAN T4_U [L1]	18,522	3,099

Index Rebuild Danger

table	rows	blks	empty	av row
TABLE_B	72,140	1,504	0	141

table	index	unique	NDV	#LB	CLUF
TABLE_B	INDEX_B_U	U	72,140	1,177	23,193

id	card	operation
0		SELECT STATEMENT ALL_ROWS
1A	1	HASH JOIN SEMI
2	1	TABLE ACCESS BY INDEX ROWID TABLE_B [A]
3A	1	INDEX RANGE SCAN INDEX_B_U [A]
4	72,141	VIEW VW_NSO_1 [VW_NSO_1]
5F		FILTER
6	72,141	HASH GROUP BY
7	72,141	VIEW [from$_subquery$_003]
8		UNION-ALL
9	1	PARTITION RANGE SINGLE :4-4
10F	1	TABLE ACCESS FULL TABLE_A [B]:4-4
11	72,140	INDEX FAST FULL SCAN TABLE_B [C]

Index Rebuild Danger

table	rows	blks	empty	av row
TABLE_B	72,140	1,504	0	141

table	index	unique	NDV	#LB	CLUF
TABLE_B	INDEX_B_U	U	2,807,924	45,804	902,749

id	card	operation
0		SELECT STATEMENT ALL_ROWS
1	1	TABLE ACCESS BY INDEX ROWID TABLE_B [A]
2AF	1	INDEX RANGE SCAN INDEX_B_U [A]
3F		FILTER
4	72,141	HASH GROUP BY
5	72,141	VIEW [from$_subquery$_003]
6		UNION-ALL
7	1	PARTITION RANGE SINGLE :6-6
8F	1	TABLE ACCESS FULL TABLE_A [B]:6-6
9	72,140	TABLE ACCESS FULL TABLE_B [C]

References

Jiang, L. *DSS Performance in Oracle Database 11g* Available from http://www.oracle.com/technology/products/bi/db/11g/pdf/twp_bidw_dss_perf_11gr1.pdf

Lewis, Jonathan. *Cost-Based Oracle: Fundamentals.* Apress. ISBN: 1590596366

Metalink 465787.1. *Managing CBO Stats During an Upgrade from 9i to 10g*

Optimizer Development Group. *Improvement of Auto Sampling Statistics Gathering Feature in Oracle 11g* [Blog]. Available from http://optimizermagic.blogspot.com/2008/01/improvement-of-auto-sampling-statistics.html.

Rahn, Greg. *Oracle 11g: Enhancements to Dbms_Stats* [Blog]. Available from http://structureddata.org/2007/09/17/oracle-11g-enhancements-to-dbms_stats/

Seiler, Don. *Dr. Statslove Or: How I Learned to Stop Guessing and Love the 10053 Trace* [Blog]. Available from http://seilerwerks.wordpress.com/2007/08/17/dr-statslove-or-how-i-learned-to-stop-guessing-and-love-the-10053-trace/

Wolfgang Breitling

breitliw@centrexcc.com

Centrex Consulting Corp.

www.centrexcc.com