

Histograms

-

Myths and Facts



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

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

Oracle since 1993 (7.0.12)

Mathematics major from University of Stuttgart



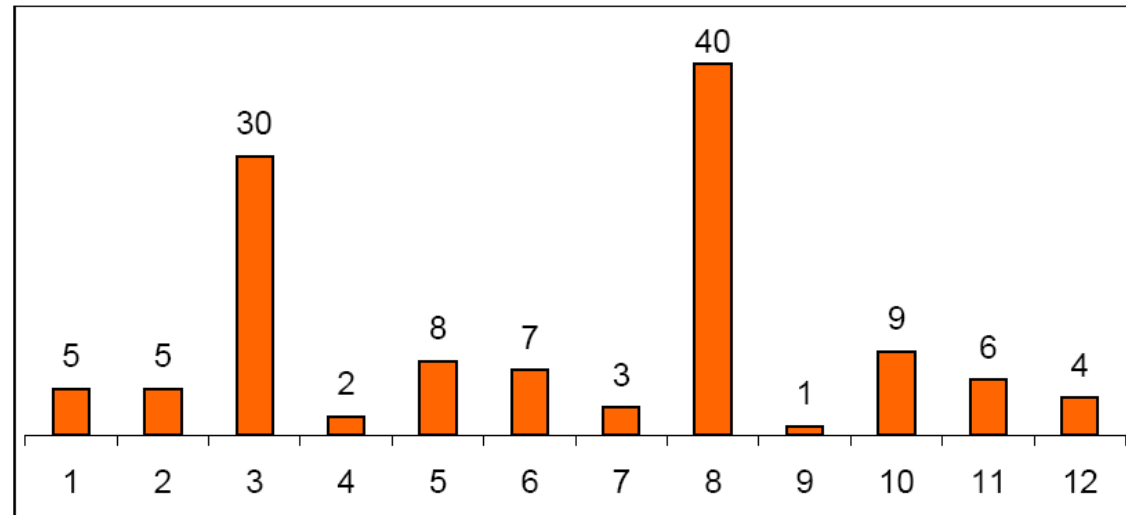
Agenda

- ❖ Look at two (still) rather widespread misconceptions about histograms
- ❖ Explore the relationship between the number of buckets and the column statistics, particularly for height-balanced histograms
- ❖ Explore the effect of some “auto-options” in statistics gathering
- ❖ Offer Guidelines for Statistics Gathering

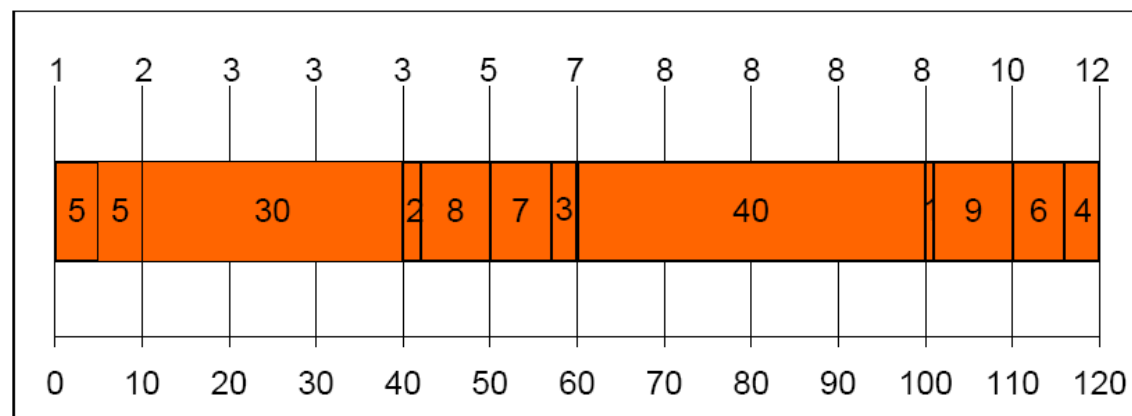


Two Types of Histograms

Frequency
or equi-width:

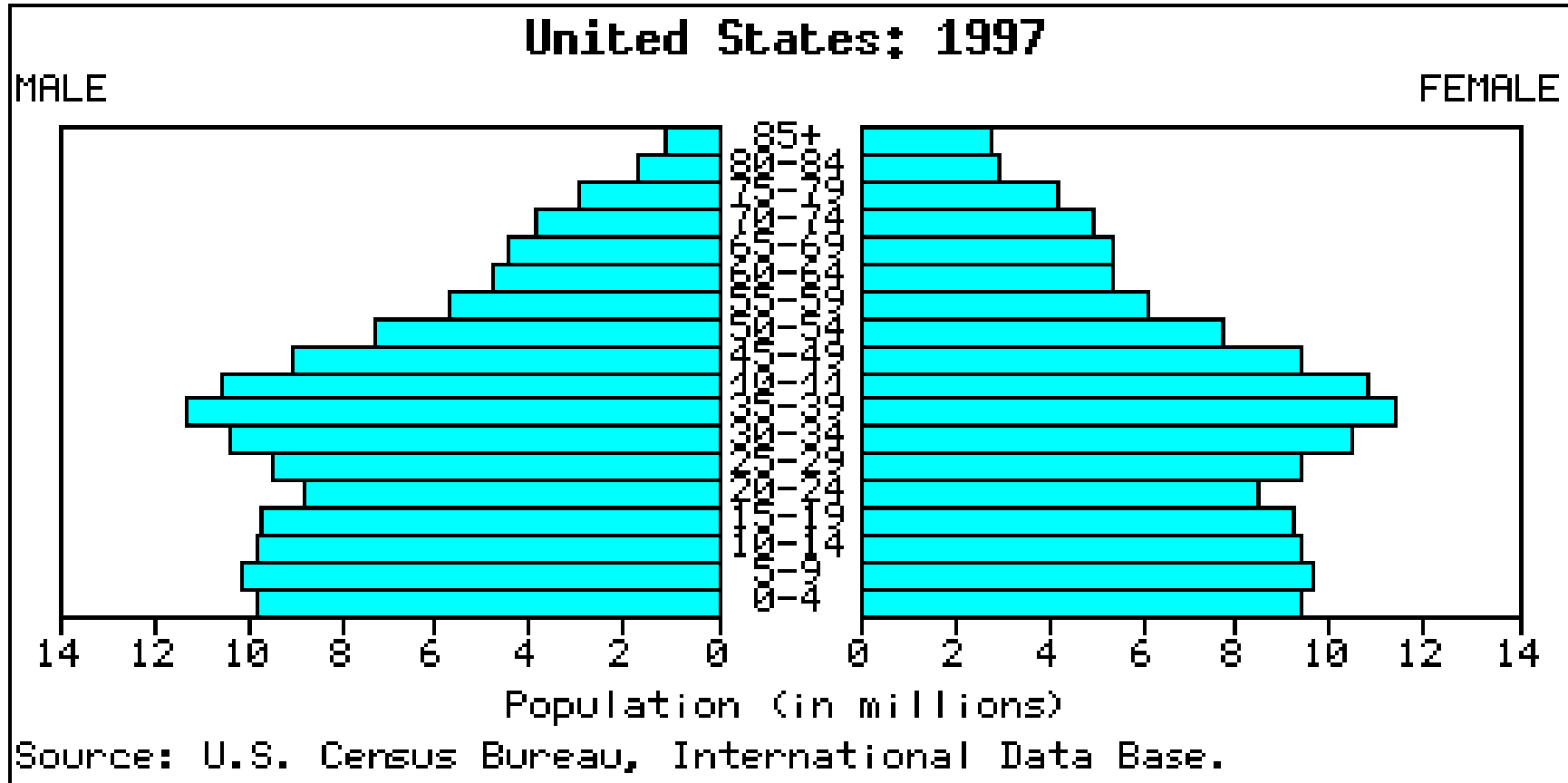


Height balanced
or equi-depth:





Age “Pyramid”





Myth or Fact ?

Histograms allow the CBO to make a better choice between index access and table scan and are therefore only needed for indexed columns



Histogram on non-indexed Column

```
create table t1 (pk1 number
, pk2 number
, fk1 number
, fk2 number
, d1 date
, d2 number
, d3 number
, d4 varchar2(2000)
);
```

```
create unique index t1p on t1(pk1, pk2);
```

```
create index t1x on t1(d2);
```

```
select sum(t1.d2*t2.d3*t3.d3)
from t1, t2, t3
where t1.fk1 = t2.pk1
and t3.pk1 = t2.fk1
and t3.d2 = 35
and t1.d3 = 0;
```



Histogram on non-indexed Column

begin

dbms_random.seed(67108863);

for i in 1..250 loop

insert into t1

select i, rownum j

, mod(trunc(100000*dbms_random.value),10)+1

, mod(trunc(100000*dbms_random.value),10)+1

, trunc(sysdate)+trunc(100*dbms_random.value)

, mod(trunc(100000*dbms_random.value),100)+1

, decode(mod(trunc(100000*dbms_random.value), 65535),0,0,1)

, mod(trunc(100000*dbms_random.value),1000)+1

, dbms_random.string('A',trunc(abs(2000*dbms_random.value)))

from dba_objects where rownum <= 250;

insert into t2 ...

insert into t3 ...



Without Histogram

<u>call</u>	<u>count</u>	<u>cpu</u>	<u>elapsed</u>	<u>disk</u>	<u>query</u>	<u>current</u>	<u>rows</u>
Parse	1	0.00	0.01	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	1.57	8.97	44109	47534	0	1
total	4	1.57	8.99	44109	47534	0	1

<u>Rows</u>	<u>Row Source Operation</u>
1	SORT AGGREGATE (cr=47534 r=44109 w=0 time=8976818 us)
1392	HASH JOIN (cr=47534 r=44109 w=0 time=8975554 us)
2	TABLE ACCESS FULL T1 (cr=23498 r=22165 w=0 time=4159263 us)
175296	HASH JOIN (cr=24036 r=21944 w=0 time=4625555 us)
627	TABLE ACCESS BY INDEX ROWID T3 (cr=632 r=0 w=0 time=3727 us)
627	INDEX RANGE SCAN T3X (cr=7 r=0 w=0 time=583 us)
62500	TABLE ACCESS FULL T2 (cr=23404 r=21944 w=0 time=4430244 us)



With Histogram

<u>call</u>	<u>count</u>	<u>cpu</u>	<u>elapsed</u>	<u>disk</u>	<u>query</u>	<u>current</u>	<u>rows</u>
Parse	1	0.00	0.08	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	0.68	4.72	22161	24313	0	1
total	4	0.68	4.81	22161	24313	0	1

<u>Rows</u>	<u>Row Source Operation</u>
1	SORT AGGREGATE (cr=24313 r=22161 w=0 time=4722357 us)
1392	HASH JOIN (cr=24313 r=22161 w=0 time=4720858 us)
627	TABLE ACCESS BY INDEX ROWID T3 (cr=632 r=0 w=0 time=589957 us)
627	INDEX RANGE SCAN T3X (cr=7 r=0 w=0 time=9110 us)
500	TABLE ACCESS BY INDEX ROWID T2 (cr=23681 r=22161 w=0 time=4109511 us)
503	NESTED LOOPS (cr=23504 r=21984 w=0 time=4045239 us)
2	TABLE ACCESS FULL T1 (cr=23498 r=21980 w=0 time=4020665 us)
500	INDEX RANGE SCAN T2P (cr=6 r=4 w=0 time=23835 us)



Without Histogram

<u>Rows</u>	<u>Card</u>	<u>Plan</u>
	1	SELECT STATEMENT (choose)(Cost 5532)
1	1	SORT (aggregate)
479	7,704,878	HASH JOIN (Cost 5532)
2	31,250	TABLE ACCESS T1 (full)(Cost 3379)
61,428	61,639	HASH JOIN (Cost 2061)
492	492	TABLE ACCESS T3 (by index rowid)(Cost 485)
492	492	INDEX NON-UNIQUE T3X (range scan)(Columns 1)(Cost 3)
62,500	62,500	TABLE ACCESS T2 (full)(Cost 1574)



With Histogram

<u>Rows</u>	<u>Card</u>	<u>Plan</u>
	1	SELECT STATEMENT (choose)(Cost 3955)
1	1	SORT (aggregate)
479	493	HASH JOIN (Cost 3955)
492	492	TABLE ACCESS T3 (by index rowid)(Cost 485)
492	250	INDEX NON-UNIQUE T3X (range scan)(Columns 1)(Cost 3)
500	250	TABLE ACCESS T2 (by index rowid)(Cost 45)
503	500	NESTED LOOPS (Cost 3469)
2	2	TABLE ACCESS T1 (full)(Cost 3379)
500	250	INDEX UNIQUE T2P (range scan)(Columns 1)(Cost 3)



Myth or Fact ?

If I have the time window,
there is no harm in collecting
histograms on all columns.

(so that I don't miss any that I really need)



Histograms on all Indexed Columns?

```
SELECT B1.SETID, B3.EMPLID , B3.EMPL_RCD, B3.EFFSEQ, B3.EFFDT b3_effdt, B3.CURRENCY_CD,
...
  From PS_GRP_DTL B1
    , PS_JOB B3
    , PS_JOBCODE_TBL B4
WHERE B1.SETID = rtrim(:b1, ' ')
      AND B3.EMPLID = rtrim(:b2, ' ')
      AND B1.SETID = B4.SETID
      AND B1.BUSINESS_UNIT IN (B3.BUSINESS_UNIT, ' ')
      AND B3.BUSINESS_UNIT IN ('BU001', 'BU007', 'BU017', 'BU018', 'BU502', 'BU101', ' ')
      AND B1.DEPTID IN (B3.DEPTID, ' ')
      AND B1.JOB_FAMILY IN (B4.JOB_FAMILY, ' ')
      AND B1.LOCATION IN (B3.LOCATION, ' ')
      AND B3.JOBCODE = B4.JOBCODE
      AND B4.EFF_STATUS = 'A'
      AND B1.EFFDT = (SELECT MAX(A1.EFFDT) FROM PS_GRP_DTL A1
                      WHERE B1.SETID = A1.SETID AND B1.JOB_FAMILY = A1.JOB_FAMILY
                      AND B1.LOCATION = A1.LOCATION AND B1.DEPTID = A1.DEPTID
                      AND A1.EFFDT <= to_date(:b3, 'yyyy-mm-dd'))
      AND B3.EFFDT = (SELECT MAX(A2.EFFDT) FROM PS_JOB A2
                      WHERE B3.EMPLID = A2.EMPLID AND B3.EMPL_RCD = A2.EMPL_RCD
                      AND A2.EFFDT <= to_date(:b3, 'yyyy-mm-dd'))
      AND B3.EFFSEQ = (SELECT MAX(A3.EFFSEQ) FROM PS_JOB A3
                      WHERE B3.EMPLID = A3.EMPLID AND B3.EMPL_RCD = A3.EMPL_RCD AND B3.EFFDT = A3.EFFDT)
      AND B4.EFFDT = (SELECT MAX(A4.EFFDT) FROM PS_JOBCODE_TBL A4
                      WHERE B4.JOBCODE = A4.JOBCODE AND A4.EFFDT <= to_date(:b3, 'yyyy-mm-dd'))
ORDER BY B1.SETID desc, B3.EMPLID desc, B1.BUSINESS_UNIT desc
      , B1.DEPTID desc, B1.JOB_FAMILY desc, B1.LOCATION desc
```



Histograms on all Indexed Columns?

without histograms

<u>call</u>	<u>count</u>	<u>cpu</u>	<u>elapsed</u>	<u>disk</u>	<u>query</u>	<u>current</u>	<u>rows</u>
Parse	1	0.01	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	0.01	0.00	0	68	0	2
total	4	0.02	0.01	0	68	0	2

with histograms on all indexed columns

<u>call</u>	<u>count</u>	<u>cpu</u>	<u>elapsed</u>	<u>disk</u>	<u>query</u>	<u>current</u>	<u>rows</u>
Parse	1	0.01	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	2	80.11	78.23	0	770842	0	2
total	4	80.12	78.24	0	770842	0	2



Myth or Fact ?

When collecting a histogram,
the greater the number of buckets
the higher the accuracy



Histograms and Density

without histogram

$$\text{density} = 1/\text{num_distinct}$$

with height balanced histogram

$$\text{density} = \sum \text{cnt}^2 / (\text{num_rows} * \sum \text{cnt})$$

with frequency histogram

$$\text{density} = 1/(2*\text{num_rows})$$



Height Balanced Histogram

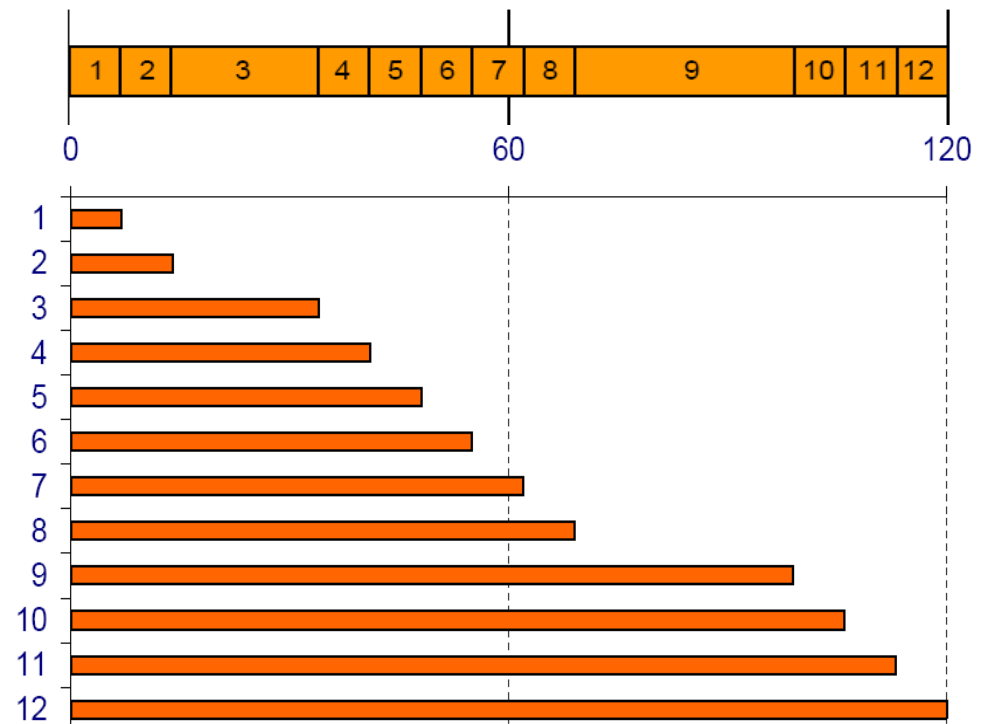
```
select min(minbkt), maxbkt
      , min(val) minval, max(val) maxval
      , sum(rep) sumrep, sum(repsq) sumrepsq
      , max(rep) maxrep, count(*) bktndv
from ( select val
      , min(bkt) minbkt
      , max(bkt) maxbkt
      , count(val) rep
      , count(val)*count(val) repsq
from ( select <column> val
      , ntile(<size>) over (order by <column>) bkt
from <owner>.<table> [partition (<partition>)] t
where <column> is not null )
group by val )
group by maxbkt
order by maxbkt
```



Height Balanced Histogram

Buckets = 2

<u>val</u>	<u>minbkt</u>	<u>maxbkt</u>	<u>rep</u>	<u>repsq</u>
1	1	1	7	49
2	1	1	7	49
3	1	1	20	400
4	1	1	7	49
5	1	1	7	49
6	1	1	7	49
7	1	2	7	49
8	2	2	7	49
9	2	2	30	900
10	2	2	7	49
11	2	2	7	49
12	2	2	7	49
			120	1790



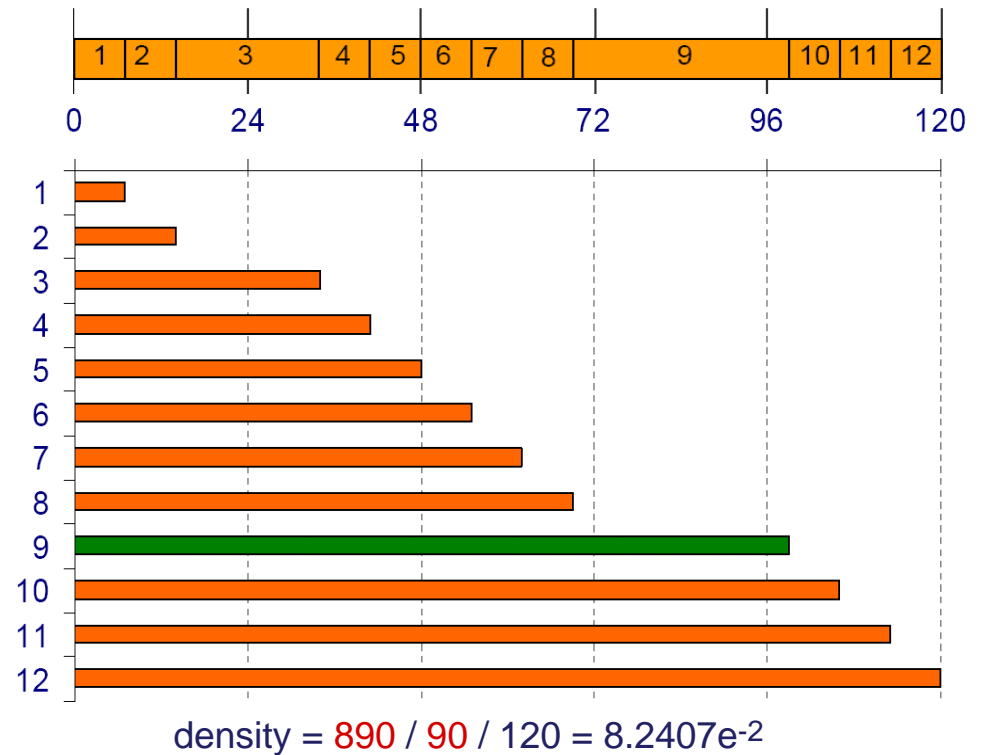
$$\text{density} = 1790 / 120 / 120 = 1.2431e^{-1}$$



Height Balanced Histogram

Buckets = 5

val	minbkt	maxbkt	rep	repsq
1	1	1	7	49
2	1	1	7	49
3	1	2	20	400
4	2	2	7	49
5	2	2	7	49
6	3	3	7	49
7	3	3	7	49
8	3	3	7	49
9	5	5	30	900
10	5	5	7	49
11	5	5	7	49
12	5	5	7	49
			90	890

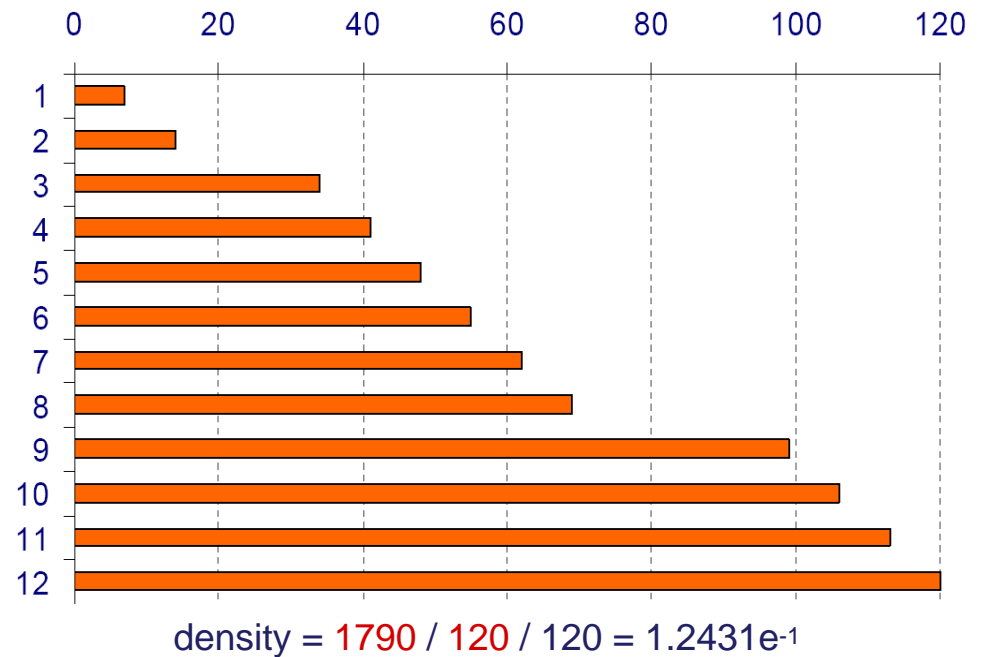




Height Balanced Histogram

Buckets = 6

val	minbkt	maxbkt	rep	repsq
1	1	1	7	49
2	1	1	7	49
3	1	2	20	400
4	2	3	7	49
5	3	3	7	49
6	3	3	7	49
7	3	4	7	49
8	4	4	7	49
9	4	5	30	900
10	5	6	7	49
11	6	6	7	49
12	6	6	7	49
			120	1790

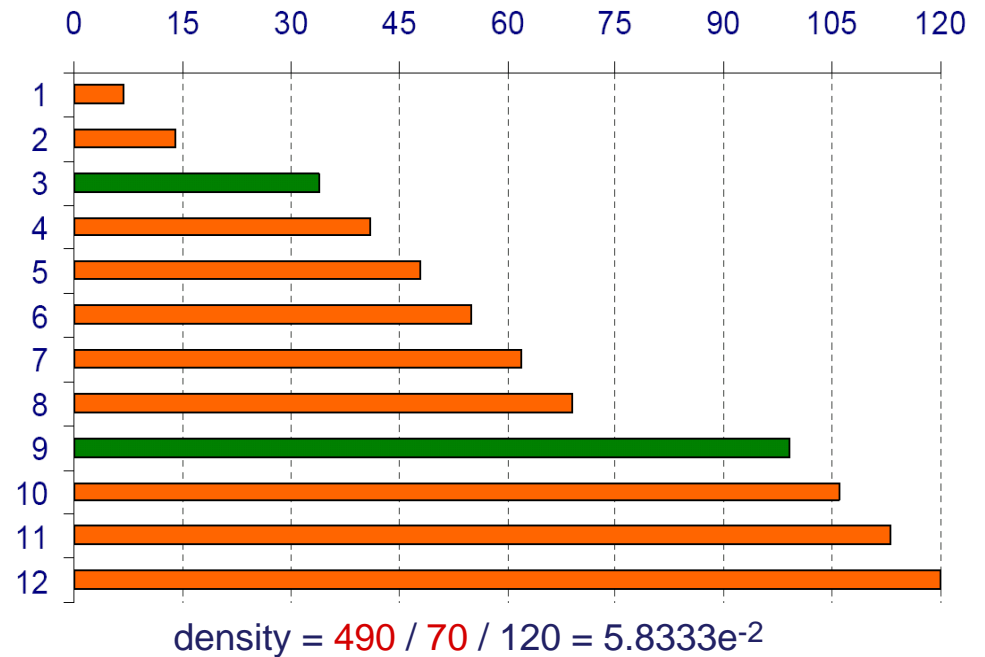




Height Balanced Histogram

Buckets = 8

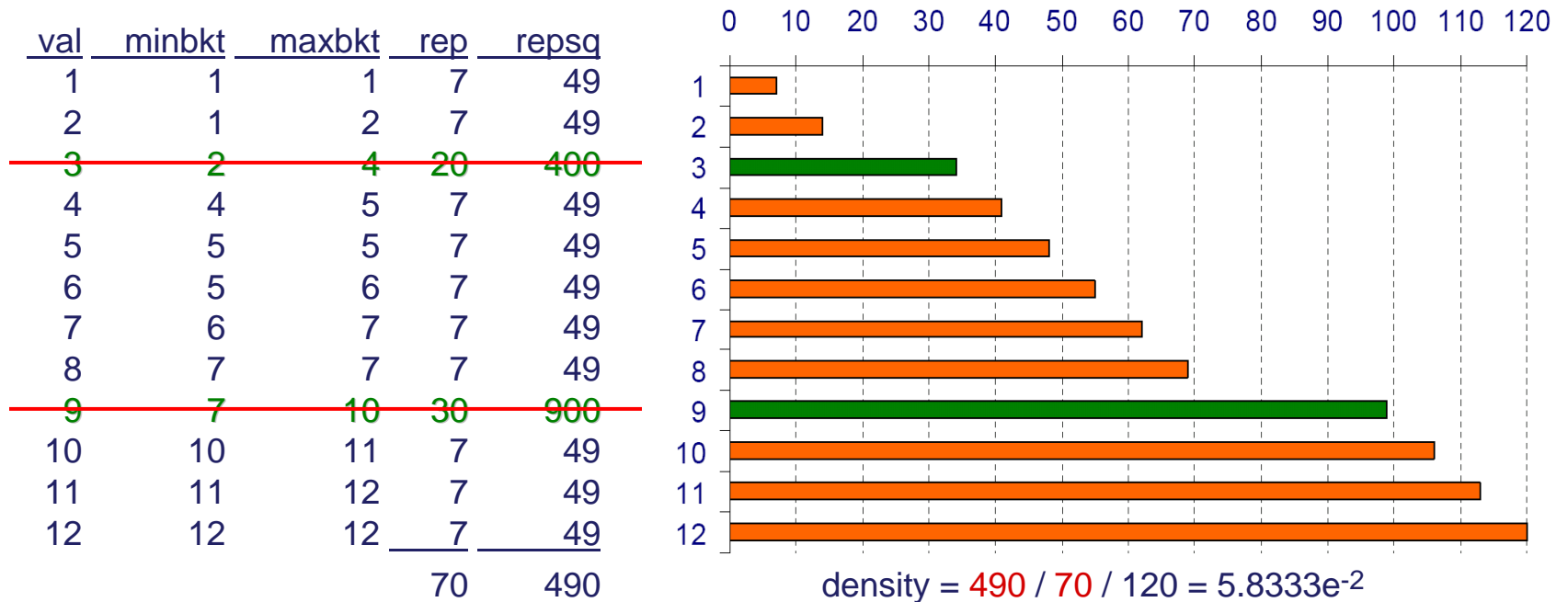
val	minbkt	maxbkt	rep	repsq
1	1	1	7	49
2	1	1	7	49
3	1	3	20	400
4	3	3	7	49
5	3	4	7	49
6	4	4	7	49
7	4	5	7	49
8	5	5	7	49
9	5	7	30	900
10	7	8	7	49
11	8	8	7	49
12	8	8	7	49
			70	490





Height Balanced Histogram

Buckets = 12

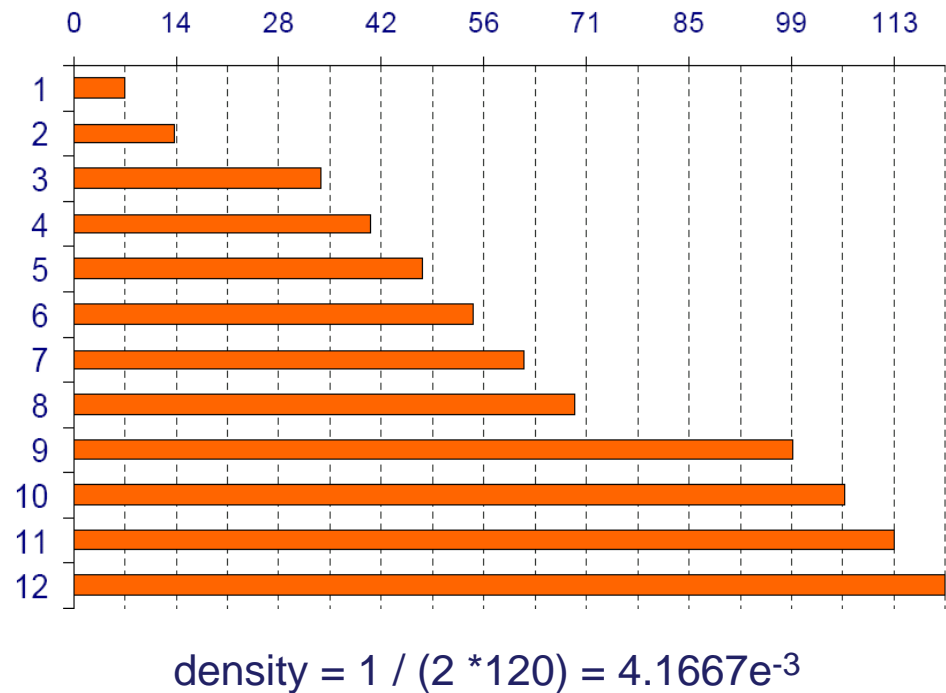




Height Balanced Histogram

Buckets = 17

<u>val</u>	<u>minbkt</u>	<u>maxbkt</u>	<u>rep</u>	<u>repsq</u>
1	1	1	7	49
2	1	2	7	49
3	2	5	20	400
4	5	6	7	49
5	6	7	7	49
6	7	8	7	49
7	8	9	7	49
8	9	10	7	49
9	10	14	30	900
10	15	15	7	49
11	16	16	7	49
12	17	17	7	49
			<u>70</u>	<u>490</u>





Size SKEWONLY ?

So how are you supposed to gather histograms and set # of buckets (size)

SKEWONLY ?

“I highly recommend that all Oracle DBA's use the `method_opt=skewonly` option to automatically identify skewed column values and generate histograms.”*

* http://www.dba-oracle.com/oracle_tips_skewonly.htm



Size SKEWONLY

SKEWONLY - Oracle determines the columns to collect histograms based on the data distribution of the columns.

Supplied PL-SQL Packages and Types Reference – `dbms_stats.gather_table_stats`

SIZE SKEWONLY when you collect histograms with the SIZE option set to SKEWONLY, we collect histogram data in memory for all specified columns (if you do not specify any, all columns are used). Once an "in-memory" histogram is computed for a column, it is stored inside the data dictionary only if it has "popular" values (multiple end-points with the same value which is what we define by "there is skew in the data").

http://asktom.oracle.com/pls/ask/f?p=4950:8:::::F4950_P8_DISPLAYID:5792247321358#7417227783861



Size SKEWONLY ?

```
create table test (n1 number not null);
```

```
insert into test (n1) select mod(rownum,5)+1 from dba_objects where  
rownum <= 100;
```

```
exec DBMS_STATS.GATHER_TABLE_STATS (null, 'test'  
 , method_opt => 'FOR ALL COLUMNS SIZE 1');
```

```
@colstats test
```

<u>table</u>	<u>column</u>	<u>NDV</u>	<u>density</u>	<u>nulls</u>	<u>lo</u>	<u>hi</u>
TEST	N1	5	2.0000E-01	0	1	5

```
@histogram test n1
```

<u>table</u>	<u>column</u>	<u>EP</u>	<u>value</u>
TEST	N1	0	1
TEST	N1	1	5



Size SKEWONLY ??

```
exec DBMS_STATS.DELETE_TABLE_STATS (null, 'test');
```

```
exec DBMS_STATS.GATHER_TABLE_STATS (null, 'test'  
    , method_opt => 'FOR COLUMNS N1 SIZE SKEWONLY');
```

```
@colstats test
```

<u>table</u>	<u>column</u>	<u>NDV</u>	<u>density</u>	<u>nulls</u>	<u>lo</u>	<u>hi</u>
TEST	N1	5	5.0000E-03	0	1	5

```
@histogram test n1
```

<u>table</u>	<u>column</u>	<u>EP</u>	<u>value</u>
TEST	N1	20	1
TEST	N1	40	2
TEST	N1	60	3
TEST	N1	80	4
TEST	N1	100	5



Size SKEWONLY ?

```
create table test (n1 number not null);
```

```
insert into test (n1) select 1 from dba_objects where rownum <= 100;
```

```
exec DBMS_STATS.GATHER_TABLE_STATS (null, 'test'  
    , method_opt => 'FOR ALL COLUMNS SIZE 1');
```

```
@colstats test
```

<u>table</u>	<u>column</u>	<u>NDV</u>	<u>density</u>	<u>nulls</u>	<u>lo</u>	<u>hi</u>
TEST	N1	1	1.0000E+00	0	1	1

```
@histogram test n1
```

<u>table</u>	<u>column</u>	<u>EP</u>	<u>value</u>
TEST	N1	0	1
TEST	N1	1	1



Size SKEWONLY ???

```
exec DBMS_STATS.DELETE_TABLE_STATS (null, 'test');
```

```
exec DBMS_STATS.GATHER_TABLE_STATS (null, 'test'  
    , method_opt => 'FOR COLUMNS N1 SIZE SKEWONLY');
```

```
@colstats test
```

<u>table</u>	<u>column</u>	<u>NDV</u>	<u>density</u>	<u>nulls</u>	<u>lo</u>	<u>hi</u>
TEST	N1	1	5.0000E-03	0	1	1

```
@histogram test n1
```

<u>table</u>	<u>column</u>	<u>EP</u>	<u>value</u>
TEST	N1	100	1



Size REPEAT ?

```
create table test( n1 number not null, n2 number not null  
    , n3 number not null, filler varchar2(4000));
```

```
exec dbms_random.seed(134217727);
```

```
insert into test
```

```
    select 100+trunc(60*dbms_random.normal),  
    100+trunc(20*dbms_random.normal),  
    decode(mod(trunc(10000*dbms_random.normal),16383),0,0,1),  
    dbms_random.string('a',2000)  
    from dba_objects  
    where rownum <= 5000;
```

```
insert into test select * from test;
```

```
insert into test select * from test;
```

```
insert into test select * from test;
```



Size REPEAT ?

```
dbms_stats.gather_table_stats (null, 'test'  
    , estimate_percent => dbms_stats.auto_sample_size);
```

<u>table</u>	<u>column</u>	<u>NDV</u>	<u>density</u>	<u>lo</u>	<u>hi</u>	<u>bkts</u>	<u>sample</u>
TEST	N1	63	1.5873E-02	66	133	1	5.5K
	N2	21	4.7619E-02	90	110	1	
	N3	2	5.0000E-01	0	1	1	
	FILLER	5000	1.9976E-04			1	

```
dbms_stats.gather_table_stats (null, 'test', estimate_percent => null  
    , method_opt => 'for columns size skewonly n1,n2,n3');
```

<u>table</u>	<u>column</u>	<u>NDV</u>	<u>density</u>	<u>lo</u>	<u>hi</u>	<u>bkts</u>	<u>sample</u>
TEST	N1	64	1.2500E-05	66	133	63	40.0K
	N2	21	1.2500E-05	90	110	20	
	N3	2	1.2500E-05	0	1	1	
	FILLER	5000	1.9976E-04			1	5.5K



Size REPEAT ?

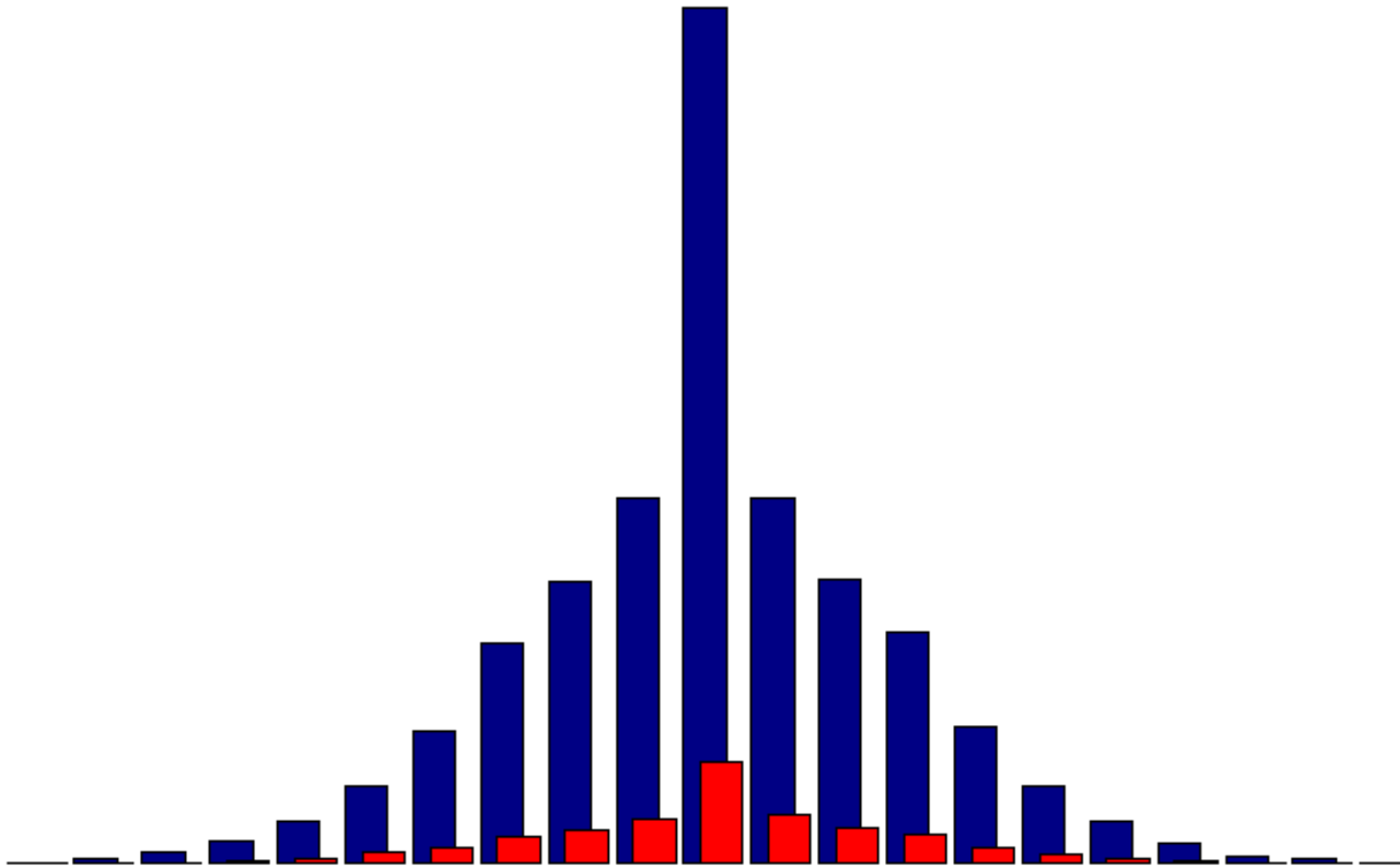
```
dbms_stats.gather_table_stats (null, 'test'  
    , estimate_percent => dbms_stats.auto_sample_size  
    , method_opt => 'for all columns size repeat');
```

table	column	NDV	density	lo	hi	bkts	sample
TEST	N1	61	1.2701E-05	66	130	60	5.0K
	N2	21	1.2701E-05	90	110	20	
	N3	2	1.2701E-05	0	1	1	
	FILLER	5000	2.0016E-04			1	

<u>table</u>	<u>column</u>	<u>EP</u>	<u>value</u>	<u>table</u>	<u>column</u>	<u>EP</u>	<u>value</u>
TEST	N3	8	0	TEST	N3	2	0
TEST	N3	40000	1	TEST	N3	4973	1



Size REPEAT ?





Statistics Gathering

- ❖ Use individual statistic gathering commands for more control
- ❖ Gather statistics on tables with a 5% sample
- ❖ Gather statistics on indexes with compute
- ❖ Add histograms where column data is known to be skewed

Statistics Gathering: Frequency and Strategy Guidelines Metalink Note 44961.1



Statistics Gathering

“Column statistics in the form of histograms are only appropriate for columns whose distribution deviates from the expected uniform distribution.”

“Given a production system with predictable, known queries, the ‘best’ execution plan for each statement is not likely to vary over time”

“Given the ‘best’ plan is unlikely to change, frequent gathering statistics has no benefit.”

“It is actually unusual for a plan to change wildly or for stats to be finely balanced. Unusual does NOT mean it never happens.”

Statistics Gathering: Frequency and Strategy Guidelines Metalink Note 44961.1



References

http://www.ixora.com.au/newsletter/2001_04.htm

44961.1 Statistics Gathering: Frequency and Strategy Guidelines

72539.1 Interpreting Histogram Information

100229.1 Indexes - Selectivity

175258.1 How to Compute Statistics on Partitioned Tables and Indexes

1031826.6 Histograms: An Overview



My favorite websites

asktom.oracle.com

(Thomas Kyte)

integrid.info

(Tanel Põder)

www.evdbt.com

(Tim Gorman)

www.go-faster.co.uk

(David Kurtz)

www.ixora.com.au

(Steve Adams)

www.jlcomp.demon.co.uk

(Jonathan Lewis)

www.juliandyke.com

(Julian Dyke)

www.hotsos.com

(Cary Millsap)

www.miracleas.dk

(Mogens Nørgaard)

www.oracledba.co.uk

(Connor McDonald)

www.oraperf.com

(Anjo Kolk)

www.orapub.com

(Craig Shallahamer)

www.scale-abilities.com

(James Morle)

Wolfgang Breitling

breitliw@centrexcc.com

Centrex Consulting Corp.

www.centrexcc.com

