



# 拨开Oracle CBO优化器迷雾,探究Histogram直方图之秘



刘相兵(Maclean Liu)  
liu.maclean@gmail.com

ORA-ALLSTARS Exadata用  
户组QQ群:23549328

# About Me

**ORACLE®**  
Certified Master  
Oracle Database 10g  
Administrator

**ORACLE®**  
Certified Master  
Oracle Database 11g  
Administrator

- **Email:**liu.maclean@gmail.com
- **Blog:**www.askmaclean.com
- **Oracle Certified Database Administrator Master 10g and 11g**
- **Over 7 years experience with Oracle DBA technology**
- **Over 8 years experience with Linux technology**
- **Member Independent Oracle Users Group**
- **Member All China Users Group**
- **Presents for advanced Oracle topics: RAC, DataGuard, Performance Tuning and Oracle Internal.**



*For the Complete Technology & Database Professional*



**MEMBER**



# About Me





# 神秘的Histogram直方图迷雾



# 预备知识:CBO术语

**NDV – number of distinct values**

**Card – cardinality**

**NULLS: Number of Nulls in Column**

**TYPE: Histogram Type**

**#BKTS: Histogram Buckets**

**UNCOMPBKTS: Histogram Uncompressed Buckets**

**ENDPTVALS: Histogram End Point Values**

**Freq 频率直方图**

**HtBal 高度平衡直方图**

**DEN: Column Density 密度**

**sel – selectivity 选择性**

详见 <http://www.askmaclean.com/archives/cbo-terms.html>

# 预备知识: histgram\$中char的存放

直方图基表histgram\$中使用16进制存放char类型，需要转换16进制为str才能看懂

转换函数可以使用hexstr 函数，下载地址：

<http://t.cn/zYkqHoi>

```
SQL> col source_char for a15
SQL> select endpoint_number, endpoint_value
  2   from dba_tab_histograms
  3   where table_name = 'DATA_SKEW'
  4     and column_name = 'SOURCE'
  5     and rownum < 2;
```

ENDPOINT_NUMBER	ENDPOINT_VALUE
----- 1236	----- 2.6591E+35

```
SQL> select endpoint_number, hexstr(endpoint_value) source_char
  2   from dba_tab_histograms
  3   where table_name = 'DATA_SKEW'
  4     and column_name = 'SOURCE'
  5     and rownum < 2;
```

ENDPOINT_NUMBER	SOURCE_CHAR
----- 1236	----- 360 Se

## 预备知识: DBMS\_STATS.AUTO\_SAMPLE\_SIZE

默认的自动采样大小参数

**DBMS\_STATS.AUTO\_SAMPLE\_SIZE**

**AUTO\_SAMPLE\_SIZE**时会优先采样**5500**

行,dbms\_stats包内部算法会评估采样的**5500**行数据是否有效,若无效则会再次采样**55000**行数据,依此类推。

为什么是**5500**这个数字?

**5500**这个数字是经过数学论证的。可以**90%**以上保证采样获得的直方图Histogram的桶buckets中数据分布式均匀。

# 啥是直方图Histogram?

**Histogram**直方图用来描绘 **columns**上的数据分布情况  
当数据分布倾斜时**Histogram**可以有效提升**cardinality** 评估的准确度

或曰：“如果数据分布极均匀，那么是否需要**Histogram**”

是的！例如绝大多数情况下单列主键 上不必要创建直方图**Histogram**(即便存在gap 😊)

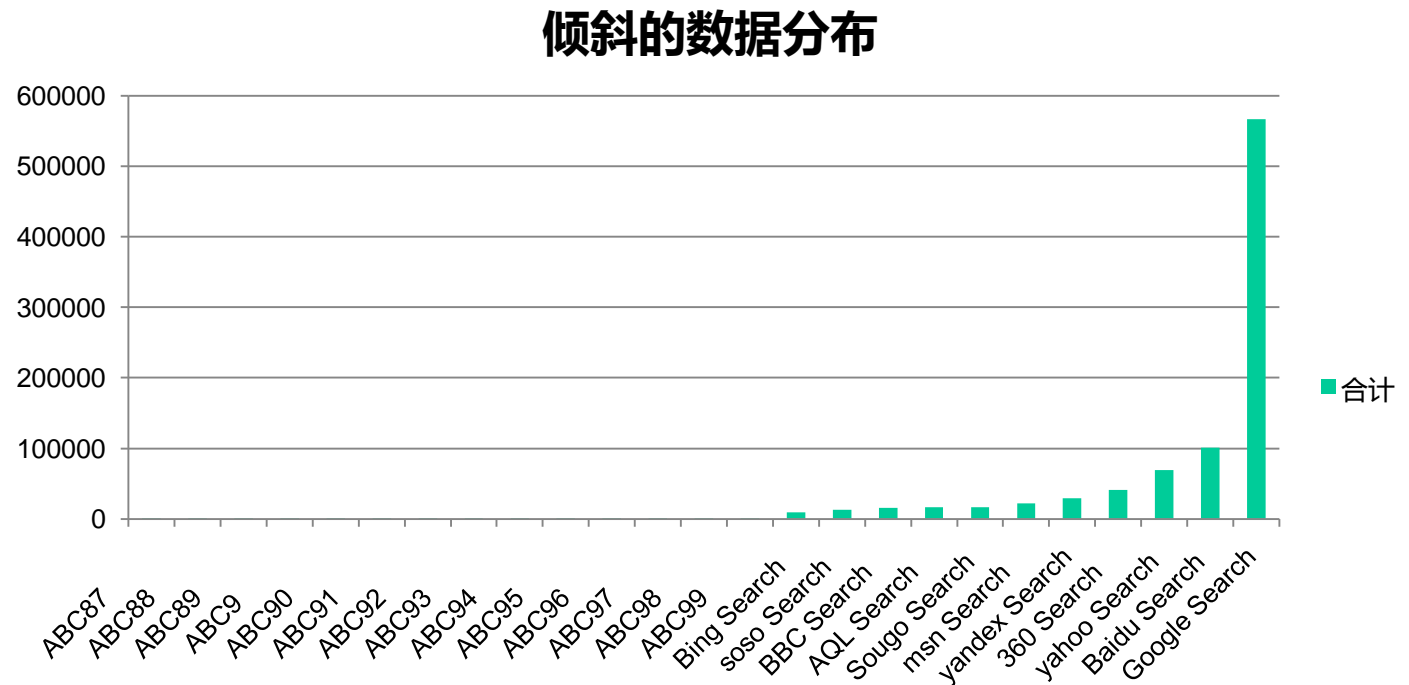
使用直方图的直接目的是为了**谓词列(column predicate)**得到更好的选择性评估



# 什么是数据倾斜？ Data Skew?

```
select source,count(*) from data_skew group by source order  
by 2,1;
```

ABC90	666
ABC91	666
ABC92	666
ABC93	666
ABC94	666
ABC95	666
ABC96	666
ABC97	666
ABC98	666
ABC99	666
Bing Search	9009
soso Search	13013
BBC Search	16016
AQL Search	17017
Sougo Search	17017
msn Search	22022
yandex Search	29029
360 Search	41041
yahoo Search	69069
Baidu Search	101101
Google Search	566566



# 数据倾斜+没有直方图啥后果？

```
SQL> exec dbms_stats.gather_table_stats(user,'DATA_SKEW',method_opt=>'FOR ALL  
COLUMNS SIZE 1'); ==》SIZE 1 不收histogram直方图
```

```
SQL> select count(*) from DATA_SKEW where source='Google Search';  
COUNT(*)
```

-----

566566

```
SQL> explain plan for select 1 from DATA_SKEW where source='Google Search';
```

```
SQL> SELECT * FROM TABLE(dbms_xplan.display);
```

-----

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

-----

0	SELECT STATEMENT		6341	82433	1256 (2)	00:00:16	
---	------------------	--	------	-------	----------	----------	--

* 1	TABLE ACCESS FULL	DATA_SKEW	<u>6341</u>	82433	1256 (2)	00:00:16	
-----	-------------------	-----------	-------------	-------	----------	----------	--

1 - filter("SOURCE"='Google Search')

优化器评估 source='Google Search'; Cardinality为 6431 ， 实际source='Google Search' 共  
566566,

差**88倍！！** 数据倾斜+没直方图==》优化器Optimizer  
很迷茫、很惆怅。。。。。

# 没有直方图，优化器怎么算Cardinality?

```
alter session set events '10053 trace name context forever,level 1';  
explain plan for select 1 from DATA_SKEW where source='Google Search';
```

## SINGLE TABLE ACCESS PATH

-----  
BEGIN Single Table Cardinality Estimation  
-----

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13.00 NDV: 158 Nulls: 0 Density: 0.0063291

Table: DATA\_SKEW Alias: DATA\_SKEW

Card: Original: 1001859 Rounded: 6341 Computed: 6340.88 Non Adjusted: 6340.88

Computed: 6340.88 → 原始计算获得的 Cardinality

Rounded: 6341 → round(Computed: 6340.88) 后的结果

非NULL比例 =  $(\text{Card: Original} - \text{Nulls}) / \text{Card: Original} = 1$

没有直方图的情况下

Sel =  $(1/\text{NDV}) * \text{非NULL比例} = (1/158) * 1 = 0.0063291$

Computed Cardinality = Card Original \* Sel =  $1001859 * 0.0063291 = 6340.88$

## 上例中如果有直方图Cardinality呢？

```
exec dbms_stats.gather_table_stats(user,'DATA_SKEW',method_opt=>'FOR ALL  
COLUMNS SIZE SKEWONLY', estimate_percent=>100);
```

```
SQL> explain plan for select 1 from DATA_SKEW where source='Google Search';
```

```
SQL> SELECT * FROM TABLE(dbms_xplan.display);
```

```
-----  
| Id | Operation          | Name      | Rows | Bytes | Cost (%CPU)| Time     |  
-----  
| 0 | SELECT STATEMENT    |           | 566K | 7192K | 1256 (2)| 00:00:16 |  
|* 1 | TABLE ACCESS FULL | DATA_SKEW | 566K | 7192K | 1256 (2)| 00:00:16 |  
-----
```

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13.00 NDV: 161 Nulls: 0 Density: 9.9500e-05

Histogram: Freq #Bkts: 161 UncompBkts: 999999 EndPtVals: 161

Table: DATA\_SKEW Alias: DATA\_SKEW

Card: Original: 999999 Rounded: 566566 Computed: 566566.00 Non Adjusted:  
566566.00

Histogram: Freq → 频率直方图      #Bkts: 161 → 161个buckets

UncompBkts → 采样获得的 ( Card Original – NULLS)



# 10053 trace中并不列出完整的直方图信息哦

直方图查看: dba\_tab\_cols、dba\_tab\_histograms、sys.histgrm\$

```
SQL> select HISTOGRAM from dba_tab_cols where table_name='DATA_SKEW' and  
column_name='SOURCE';
```

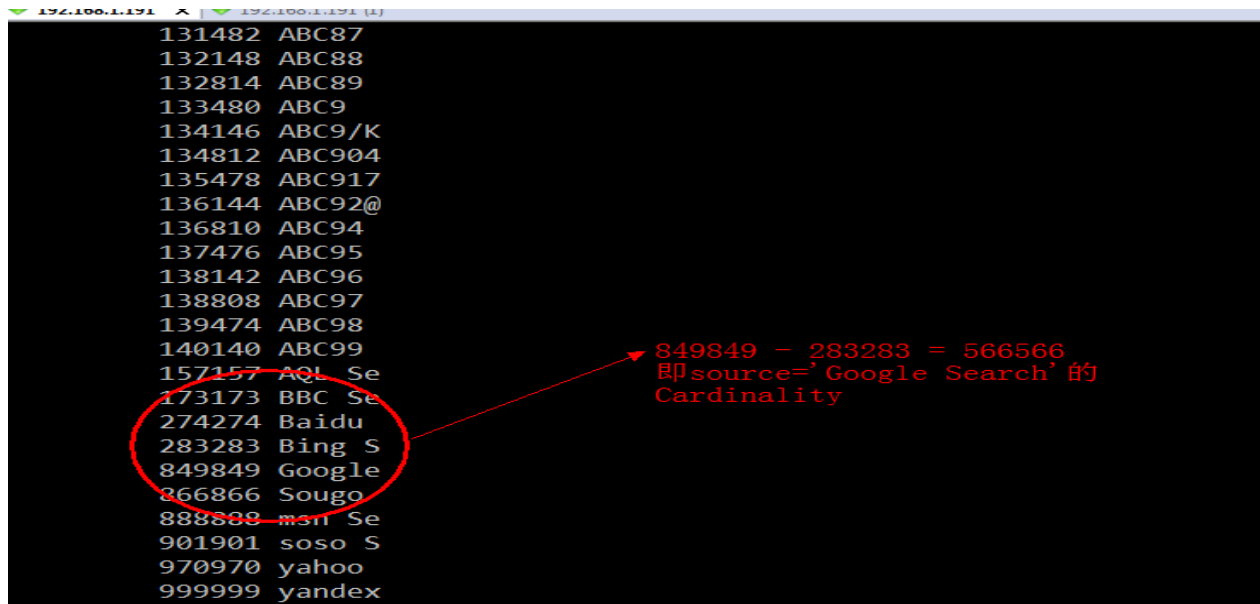
HISTOGRAM

-----

FREQUENCY                      → 频率直方图

SOURCE 是字符类型 需要 hexstr(endpoint\_value)才能看懂

```
select endpoint_number, hexstr(endpoint_value) source_char from dba_tab_histograms  
where table_name='DATA_SKEW' and column_name='SOURCE';
```



# 拨开云雾见月明—开始探究 Histogram直方图之旅



# Histogram直方图的历史

最早追述到Oracle 7版本开始引入Histogram的思想

Oracle 7中使用CBO时选择性评估很不准确

在Oracle 8中引入了Histogram作为optimizer statistics新特性，当时Frequency Histograms被称作 "value-based" histogram

## Height-Based Histograms

Height-based histograms place approximately the same number of values into each range, so that the endpoints of the range are determined by how many values are in that range.

Consider that a table's query results in the following four sample values: 4, 18, 30, and 35.

For a height-based histogram, we consider each of these values to occupy a portion of one bucket, in proportion to their size. The resulting selectivity is computed with the following formula:

$$S = \text{Height}(35) / \text{Height}(4 + 18 + 30 + 35)$$

## Value-Based Histograms

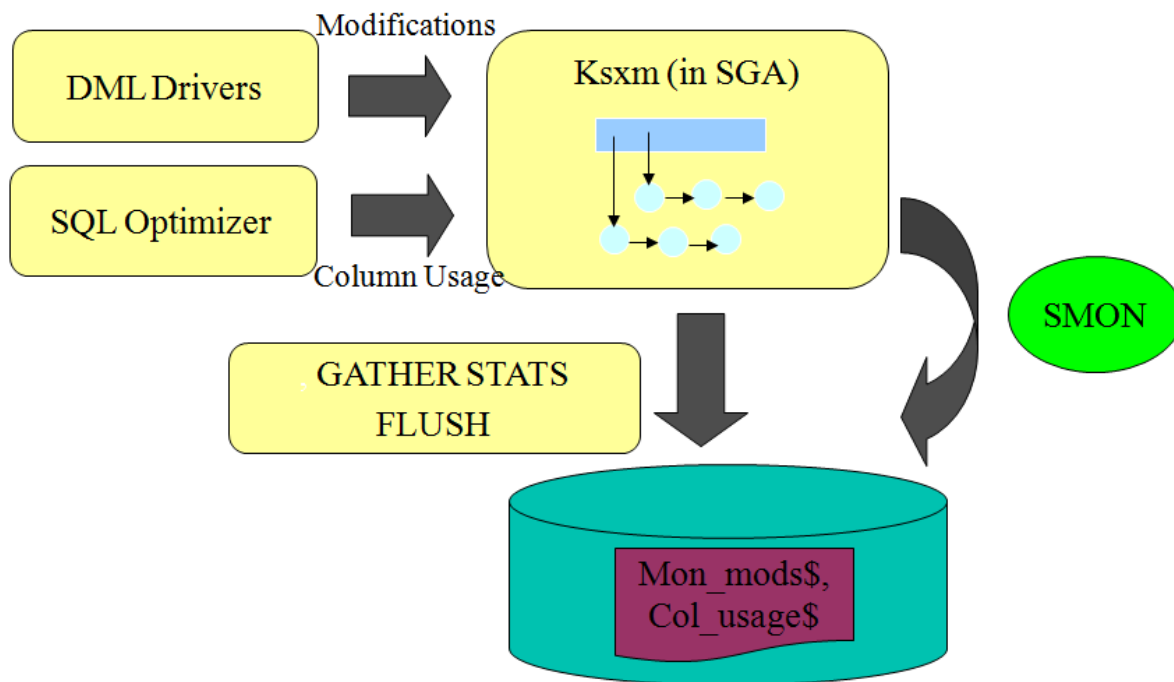
Consider the same four sample values in the example above. In a value-based histogram a bucket is used to represent each of the four distinct values. In other words, one bucket represents 4, one bucket represents 18, another represents 30, and another represents 35. The resulting selectivity is computed with the following formula:

$$S = [\text{\#rows}(35) / (\text{\#rows}(4) + \text{\#rows}(18) + \text{\#rows}(30) + \text{\#rows}(35))] / \text{\#buckets}$$

# Histogram直方图的历史

版本8~9i默认不收集直方图Histogram，需要手动指定  
**method\_opt size** (默认size=1 仅收集最小/大值，  
**distinct**值等信息)

从版本10g开始Oracle会自动收集Histogram，  
Histogram是否收集取决于col\_usage\$中记录的关于  
该列用作SQL中谓词条件的信息和数据分布情况



SMON定期将shared pool中的  
谓词使用状况刷新到  
col\_usage\$表中

例如:  
Select \* from tab where  
colA=1;

则记录为对colA充当  
EQUALITY\_PRENDS →  
equality predicates等式谓词



-

## Optimizer优化器在哪里使用Histogram?

- 有2个地方Oracle Optimizer优化器会用到Histogram:
  - 过滤谓词的选择性评估
  - Join连接基数(Cardinality)评估
- 在做Join连接基数(Cardinality)评估时，往往差之毫厘谬之千里：
  - 优化器可能选择错误的Join连接方式或顺序。
- 例如分页排序查询 因为基数评估误差导致去用了HASH JOIN.....

# Histogram的Buckets桶数

- 对于绝大多数情况默认的**75**个桶总是合适的
- 最大桶数= 最小值(**254**,其他因素限制桶数)
- 若频繁出现地**distinct**值的数据并不多，则将桶数设置为大于这个数目往往是有益的。
- 版本**12c**之前直方图最大桶数为**254**,**12c**之后.....

# Histogram的其他特性

- 早期版本中远程**DBLINK**查询时表上的直方图可能不被使用
- 直方图是静态的，当数据更新频繁时可能变得“陈旧”
- 对于超过**32**个字符的字符型列，超出的那一部分无法在直方图中体现。例如**rpadd('A',100,'Z')**和**rpadd('A',101,'Z')**被认为是同一个**value**
- 数字和日期在直方图上被精确表示
- **Popular and non-popular**值被用做帮助判断选择性



# 如何收集直方图

- 推荐使用**dbms\_stats**收集直方图，虽然**analyze**命令也能收集直方图

```
EXECUTE DBMS_STATS.GATHER_TABLE_STATS  
( 'scott', 'emp', METHOD_OPT => 'FOR COLUMNS SIZE 20 enable' );
```

- **Method\_opt**参数

```
FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause]  
FOR COLUMNS [size clause] column [size_clause] [,column...]
```

- **Size**指定直方图的桶数

```
SIZE {integer | REPEAT | AUTO | SKEWONLY}  
Integer - 人工指定直方图桶数 从1~254  
REPEAT - 刷新现有的直方图列上的信息  
AUTO - 基于数据分布和负载收集直方图  
SKEWONLY- 基于数据分布收集直方图
```

- 若对直方图不甚了解，推荐使用**AUTO**或**SKEWONLY**

# Histogram直方图的种类

版本12c之前有2种类型Histogram

- **Height Balanced Histogram**→高度平衡直方图
  - 列值被分成多个**buckets**
  - 每个**bucket**包含大致一样数目的行数
  - 当**NDV>254**时会采用高度平衡直方图(注意**dbms\_stats**采样到的**NDV**未必是实际的**NDV**)
- **Frequency Histogram(Value-Based)**→频率直方图
  - 该列上每一个值都会具有频率信息
  - 当**NDV( number of distinct values)**的个数**<=**最大桶数**buckets 254**个时使用频率直方图
- 在**Oracle**数据字典基表**sys. histgrm\$** 上这2种类型Histogram的存放方式一样

# Height Balanced Histogram

- 每个**buckets**桶里的行数都大致相同，除了最后一个桶
- 最后一个桶中的可能比其他桶中的少
- 每个桶中最大值成为**bucket value**→ **endpoint\_value**
- 每个**value**值占有一个桶的一部分，按比例

select 1 from DATA\_SKEW\_HB where source='Google Search';

Column (#2):

NewDensity:0.000403, OldDensity:0.000663 BktCnt:254, PopBktCnt:229, PopValCnt:11, NDV:255

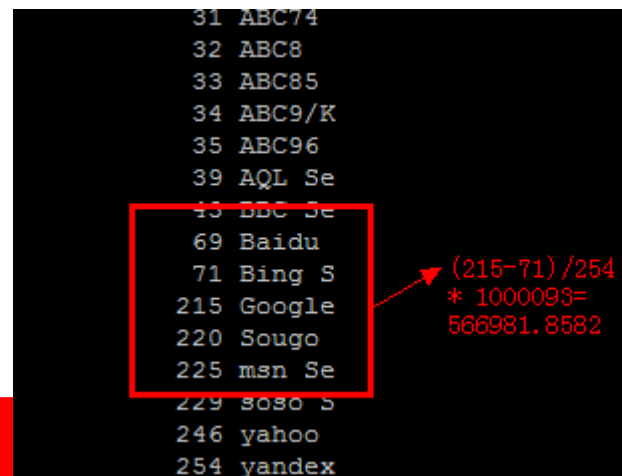
Column (#2): SOURCE(

AvgLen: 13 NDV: 255 Nulls: 0 Density: 0.000403

Histogram: HtBal #Bkts: 254 UncompBkts: 254 EndPtVals: 36

Table: DATA\_SKEW\_HB Alias: DATA\_SKEW\_HB

Card: Original: 1000093.000000 Rounded: 566982 Computed: **566981.86** Non Adjusted:  
566981.86



# Height Balanced Histogram

```
SQL> select count(distinct source) from data_skew_hb;  
COUNT(DISTINCTSOURCE)
```

-----  
255

```
SQL> exec dbms_stats.gather_table_stats(user,'DATA_SKEW_HB',method_opt=>'FOR ALL  
COLUMNS SIZE SKEWONLY',estimate_percent=>100);
```

```
SQL> select histogram from dba_tab_cols where table_name='DATA_SKEW_HB' and  
column_name='SOURCE';
```

HISTOGRAM

-----  
HEIGHT BALANCED

```
SQL> select count(*) from dba_tab_histograms where table_name='DATA_SKEW_HB' and  
column_name='SOURCE';  
COUNT(*)
```

-----  
36

```
select endpoint_number,hexstr(endpoint_value) from dba_tab_histograms where  
table_name='DATA_SKEW_HB' and column_name='SOURCE' order by 1,2
```

```
10 360 Se  
11 ABC100  
12 ABC105  
13 ABC110  
14 ABC117  
15 ABC121  
16 ABC128  
17 ABC133  
18 ABC138  
19 ABC143  
20 ABC15  
21 ABC2/b  
22 ABC26  
23 ABC30□  
24 ABC37  
25 ABC41U  
26 ABC48  
27 ABC53  
28 ABC59  
29 ABC64  
30 ABC69  
31 ABC74  
32 ABC8  
33 ABC85  
34 ABC9/K  
35 ABC96  
39 AQL Se  
43 BBC Se  
69 Baidu  
71 Bing S  
215 Google  
220 Sougo  
225 msn Se  
229 soso S
```



# Height Balanced Histogram

- 每个**buckets**桶里的行数都大致相同，除了最后一个桶
- 最后一个桶中的行数可能比其他桶中的少
- 每个桶中最大值成为**bucket value**→ **endpoint\_value**
- 每个**value**值占有一个桶的一部分，按比例

select 1 from DATA\_SKEW\_HB where source='Google Search';

Column (#2):

NewDensity:0.000403, OldDensity:0.000663 BktCnt:254, PopBktCnt:229, PopValCnt:11, NDV:255

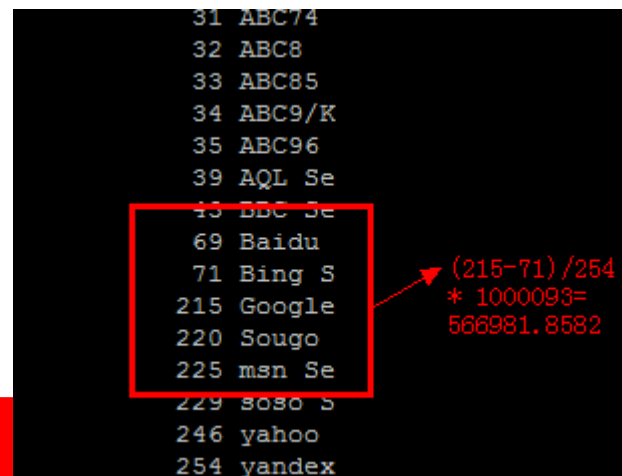
Column (#2): SOURCE(

AvgLen: 13 NDV: 255 Nulls: 0 Density: 0.000403

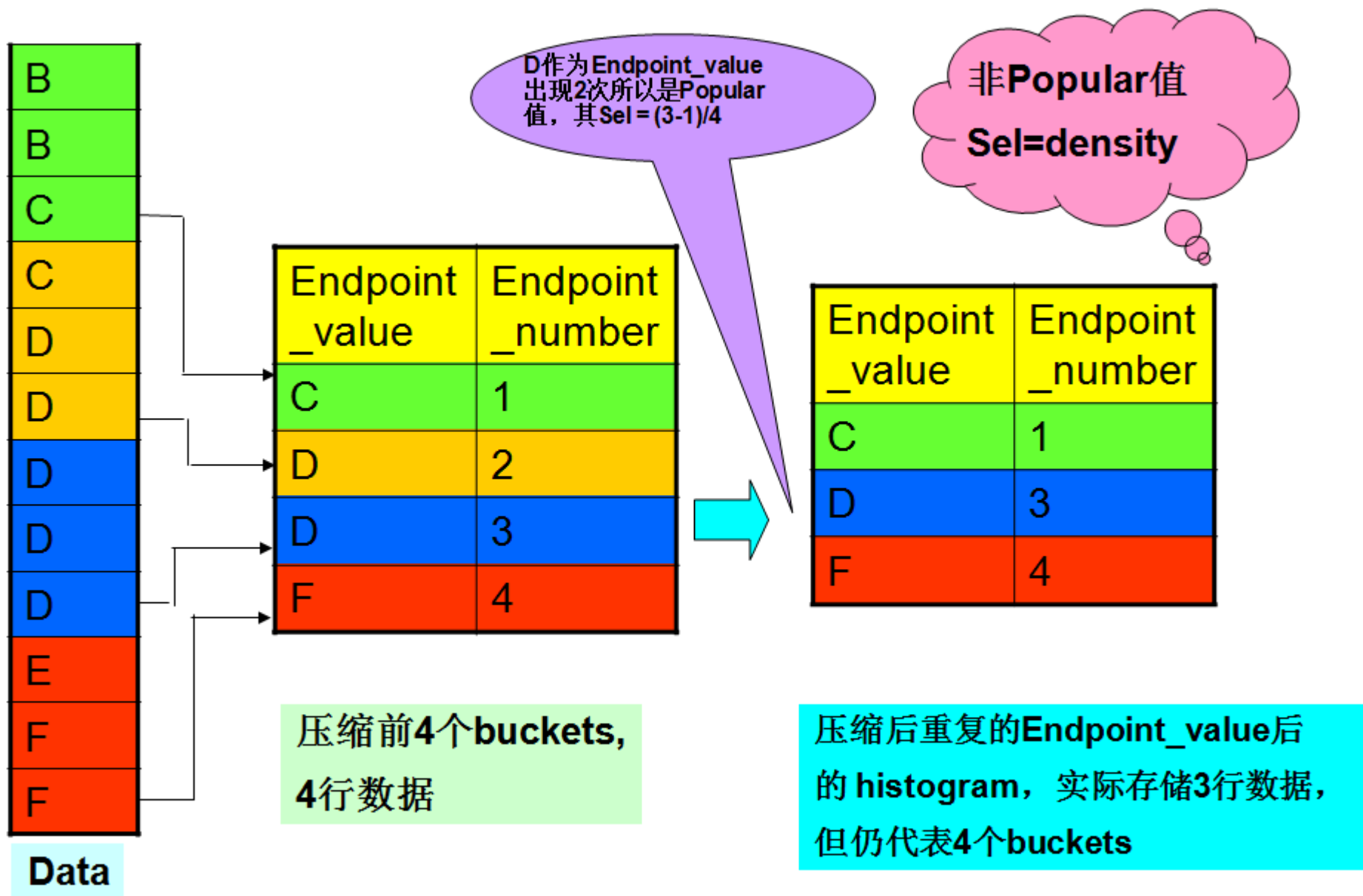
Histogram: HtBal #Bkts: 254 UncompBkts: 254 EndPtVals: 36

Table: DATA\_SKEW\_HB Alias: DATA\_SKEW\_HB

Card: Original: 1000093.000000 Rounded: 566982 Computed: **566981.86** Non Adjusted:  
566981.86



## Height Balanced Histogram → 压缩Bucket



## 10053 trace中的UNCOMPBKTS和ENDPTVALS

- 当Histogram直方图类型为frequency histograms (Histogram: Freq)时UncompBkts 等于统计信息中采样行数-NULLS, 而EndPtVals 等于bucket总数, 或者说NDV, 因为frequency histograms中 NDV=number of buckets
- 当直方图类型为height balanced histograms (Histogram: HtBal) UncompBkts 等于bucket的数目(其实也等于10053 trace中#Bkts的数目),而EndPtVals 等于已经被压缩的Histogram的大小, 换句话说等于: `select count(*) from dba_tab_histograms where table_name='&TAB' and column_name='&COL'`的实际总和。通过这2个值对比, 可以了解到popular值的多少以及数据的倾斜度, 是有很多个大量重复的值(popular value)还是仅有几个巨大的重复值。

## Height Balanced Histogram- popular value

重复出现为endpoint\_value的值称为popular value, 其Cardinality计算公式为:

$$\text{Comp\_Card} = \text{Orig\_Card} * \text{Sel}$$

$$\text{Sel} = (\text{该Popular值的桶数} / \text{总的桶数}) * \text{非NULL比例}$$

$$\text{非NULL比例} = (\text{Orig\_Card} - \text{NNulls}) / \text{Orig\_Card}$$

如上例中source='Google Search'是一个popular值:

$$\text{该Popular值的桶数} = 215 - 71 = 144$$

$$\text{非NULL比例} = (1000093 - 0) / 1000093 = 1$$

$$\text{则sel} = 144 / 254$$

$$\text{Comp\_Card} = 1000093 * 144 / 254 = 566981.86$$

## Height Balanced Histogram- non popular value

- 没有重复出现为**endpoint\_value**、或者没有充当过**endpoint\_value**的值称为**non popular value**，其**Cardinality**计算公式为：
- **Comp\_Card = Orig\_Card \* Sel**
- **Sel= Density \* 非空比例**
- 10.2.0.4以后的 **New Density= (总的buckets数- 所有popular值的buckets数PopBktCnt)/总的buckets数/ (NDV-popular值总的个数POPVALCNT)**

## Height Balanced Histogram- non popular value

```
select count(*) from data_skew_hb where source='ABC100';
```

**COUNT(\*)**

-----

**666**

**10g中10053 trace默认不显示PopBktCnt和POPVALCNT 但是可以利用后面的脚本获得**

**11g 10053 level 1 trace :**

**NewDensity:0.000403, OldDensity:0.000663 BktCnt:254, PopBktCnt:229,  
PopValCnt:11, NDV:255**

**Column (#2): SOURCE(**

**AvgLen: 13 NDV: 255 Nulls: 0 Density: 0.000403**

**Histogram: HtBal #Bkts: 254 UncompBkts: 254 EndPtVals: 36**

**Table: DATA\_SKEW\_HB Alias: DATA\_SKEW\_HB**

**Card: Original: 1000093.000000 Rounded: 403 Computed: 403.42 Non Adjusted:  
403.42**



# POPVALCNT

```
select count(*) PopValCnt from (select  
  endpoint_number,hexstr(endpoint_value) endstr,  
  endpoint_number- lag( endpoint_number,1,0) over (order by  
  endpoint_number ) as buckets from dba_tab_histograms where  
  table_name='DATA_SKEW_HB' and column_name='SOURCE')  
end_diff where buckets>1;
```

POPVALCNT

-----

11

# PopBktCnt

```
select sum(buckets) PopValCnt from (select
  endpoint_number,hexstr(endpoint_value) endstr,
  endpoint_number- lag( endpoint_number,1,0) over (order by
  endpoint_number ) as buckets from dba_tab_histograms where
  table_name='DATA_SKEW_HB' and column_name='SOURCE')
end_diff where buckets>1;
```

POPBktCNT

-----

229

# Popular value的列表

```
select endstr popvalue,buckets from (select  
  endpoint_number,hexstr(endpoint_value) endstr,  
  endpoint_number- lag( endpoint_number,1,0) over (order by  
  endpoint_number ) as buckets from dba_tab_histograms where  
  table_name='DATA_SKEW_HB' and column_name='SOURCE')  
end_diff where buckets>1;
```

POPVALUE	BUCKETS
360 Se	10
AQL Se	4
BBC Se	4
Baidu	26
Bing S	2
Google	144
Sougo	5
msn Se	5
soso S	4
yahoo	17
yandex	8

11 rows selected.

# High Balanced Histogram 计算 Density

10.2.0.4 以后 New Density = (总的buckets数 - 所有popular值的buckets数 PopBktCnt) / 总的buckets数 / (NDV - popular值总的个数 POPVALCNT)

$$= (254 - 229) / 254 / (255 - 11) = 25 / 255 / 244 = 0.000403$$

$$\text{Comp\_Card} = \text{Orig\_Card} * \text{Sel} = 1000093 * 0.000403 = 403$$

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	13	946 (2)	00:00:12
1	SORT AGGREGATE		1	13		
* 2	TABLE ACCESS FULL	DATA_SKEW_HB	403	5239	946 (2)	00:00:12

# Frequency Histogram频率直方图

- 每一个**bucket**桶代表一个列值
- 列上的所有的值均有对应的桶
- 当**NDV(采样到的)**<**min(指定的buckets数量,254)**时创建频率直方图
- **Density = 1 / ( 2 \* NumRows \* A4Nulls)**

## Frequency Histogram频率直方图-等式谓词

```
Exec dbms_stats.gather_table_stats(user,'DATA_SKEW',estimate_percent=>100);
```

```
SQL> select histogram from dba_tab_cols where table_name='DATA_SKEW' and  
       column_name='SOURCE';
```

HISTOGRAM

-----

FREQUENCY

```
SQL> select count(*) from data_skew where source='Bing Search';
```

COUNT(\*)

-----

9009

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13.00 NDV: 161 Nulls: 0 Density: 9.9500e-05

Histogram: Freq #Bkts: 161 UncompBkts: 999999 EndPtVals: 161

Table: DATA\_SKEW Alias: DATA\_SKEW

Card: Original: 999999 Rounded: 9009 Computed: **9009.00** Non Adjusted: 9009.00



# Frequency Histogram 频率直方图-等式谓词

```
select endpoint_number,  
       hexstr(endpoint_value) endstr,  
       endpoint_number - lag(endpoint_number, 1, 0) over(order by endpoint_number)  
       as rows_cnt  
from dba_tab_histograms  
where table_name = 'DATA_SKEW'  
and column_name = 'SOURCE';
```

136144	ABC92@	666
136810	ABC94	666
137476	ABC95	666
138142	ABC96	666
138808	ABC97	666
139474	ABC98	666
140140	ABC99	666
157157	AQL Se	17017
173173	BBC Se	16016
274274	Baidu	101101
283283	Bing S	9009
849849	Google	566566
866866	Sougo	17017
888888	msn Se	22022
901901	soso S	13013
970970	yahoo	69069
999999	yandex	29029

283283 - 274274 = 9009

## Frequency Histogram 频率直方图-非闭包范围

```
SQL> select count(*) from data_skew where source>= 'Sougo Search';
```

```
COUNT(*)
```

```
-----  
150150
```

136144	ABC92@	666
136810	ABC94	666
137476	ABC95	666
138142	ABC96	666
138808	ABC97	666
139474	ABC98	666
140140	ABC99	666
157157	AQL Se	17017
173173	BBC Se	16016
274274	Baidu	101101
283283	Bing S	9009
849849	Google	566566
866866	Sougo	17017
888888	msn Se	22022
901901	soso S	13013
970970	yahoo	69069
999999	yandex	29029

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13.00 NDV: 161 Nulls: 0 Density:  
9.9500e-05

Histogram: Freq #Bkts: 161 UncompBkts:  
999999 EndPtVals: 161

Table: DATA\_SKEW Alias: DATA\_SKEW

Card: Original: 999999 Rounded: 150150

Computed: 150149.50 Non Adjusted: 150149.50

**Sel= Sum(Bucketsize)/Numrows  
= 150150/ 999999**

## Frequency Histogram 频率直方图-闭包范围

SQL> select count(\*) from data\_skew where source between 'ABC904' and 'ABC94';

COUNT(\*)

2664

ABC89	666
ABC9	666
ABC90	666
ABC91	666
ABC92	666
ABC93	666
ABC94	666
ABC95	666
ABC96	666
ABC97	666
ABC98	666
ABC99	666
Bing Search	9009
soso Search	13013
BBC Search	16016
AQL Search	17017
Sougo Search	17017
msn Search	22022
yandex Search	29029

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13.00 NDV: 161 Nulls: 0 Density: 9.9500e-05

Histogram: Freq #Bkts: 161 UncompBkts: 999999 EndPtVals: 161

Table: DATA\_SKEW Alias: DATA\_SKEW

Card: Original: 999999 Rounded: 2664

Computed: 2664.00 Non Adjusted: 2664.00

**Sel= Sum(Bucketsize)/Numrows**  
**= (666+666+666+666)/ 999999**

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

- [www.askmaclean.com](http://www.askmaclean.com)

# 目前版本Histogram的欠缺

Height-Balanced Histogram中 non popular值使用density评估选择性，对于那些出现较频繁，但仍排不上popular榜单的值，使用density的评估很难准确

```
SQL> select count(*) from DATA_SKEW_HB where  
source='Macleans Search';
```

POPVALUE	BUCKETS
-----	-----
360 Se	10
AQL Se	4
BBC Se	4
Baidu	26
Bing S	2
Google	144
Sougo	5
msn Se	5
soso S	4
yahoo	17
yandex	8

```
COUNT(*)
```

```
-----  
2000
```

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13.00 NDV: 256 Nulls: 0 Density: 4.0174e-04

Histogram: HtBal #Bkts: 254 UncompBkts: 254 EndPtVals: 36

Table: DATA\_SKEW\_HB Alias: DATA\_SKEW\_HB

Card: Original: 1002093 **Rounded: 403 Computed: 402.58**

Non Adjusted: 402.58

# Histogram的未来-12c Top Frequency histograms

```
SQL> select banner from v$version where rownum=1;
```

Oracle Database 12c Enterprise Edition Release 12.1.0.0.2 - 64bit Beta

```
SQL> exec dbms_stats.gather_table_stats(user,'DATA_SKEW_HB');
```

```
SQL> select histogram from dba_tab_cols where table_name='DATA_SKEW_HB' and  
       column_name='SOURCE';
```

HISTOGRAM

-----

**TOP-FREQUENCY**

```
select count(*) from DATA_SKEW_HB where source='Maclean Search';
```

Column (#2):

NewDensity:0.000001, OldDensity:0.000000 BktCnt:1002091.000000, PopBktCnt:1001999.000000,  
PopValCnt:162, NDV:256

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13 NDV: 256 Nulls: 0 Density: 0.000001

Histogram: Top-Freq #Bkts: 1002091 UncompBkts: 1002091 EndPtVals: 254 ActualVal: yes

Table: DATA\_SKEW\_HB Alias: DATA\_SKEW\_HB

Card: Original: 1002093.000000 **Rounded: 2000 Computed: 2000.00** Non Adjusted: 2000.00

# Histogram的未来-12c Hybrid Histograms

在data\_skew\_hb表上生成更多NDV

```
SQL> exec dbms_stats.gather_table_stats(user,'DATA_SKEW_HB');
```

```
SQL> select histogram from dba_tab_cols where table_name='DATA_SKEW_HB' and  
       column_name='SOURCE';
```

HISTOGRAM

-----

**HYBRID**

```
SQL> select count(*) from DATA_SKEW_HB where source='Maclean Search';
```

COUNT(\*)

-----

**2000**

Column (#2):

NewDensity:0.000085, OldDensity:0.000000 BktCnt:5405.000000, PopBktCnt:671.000000,  
PopValCnt:5, NDV:10255

Column (#2): SOURCE(VARCHAR2)

AvgLen: 13 NDV: 10255 Nulls: 0 Density: 0.000085

**Histogram: Hybrid #Bkts: 254 UncompBkts: 5405 EndPtVals: 254 ActualVal: yes**

**Table: DATA\_SKEW\_HB Alias: DATA\_SKEW\_HB**

**Card: Original: 6001593.000000 Rounded: 2221 Computed: 2220.76 Non Adjusted: 2220.76**



你离彻底搞懂**10053 trace**这天书又近了一步



敬请期待，  
下一讲 揭秘  
Oracle优化  
器内幕，一  
次性彻底搞  
明白10053  
trace !!

更多信息

www.askmaclean.com



or

<http://www.askmaclean.com/archives/tag/tuning>

# Question & Answer



**If you have more questions later, feel free to ask**