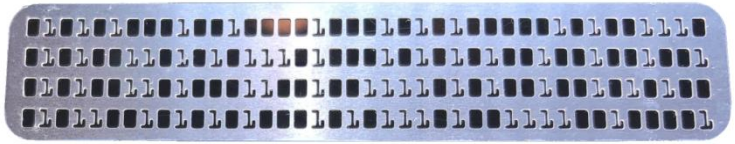
HISTOGRAMS ARE EVIL LIKE CHOCOLATE IS EVIL

Neil Chandler,
Chandler Systems

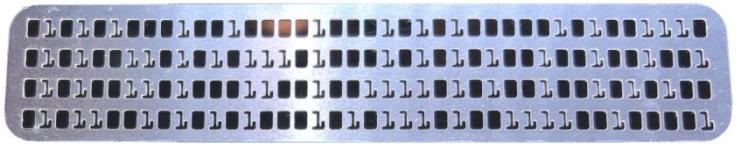


Presentation valid for Oracle 11G, 12C, 19C
Mostly valid for 23ai

HISTOGRAMS ARE EVIL LIKE CHOCOLATE IS EVIL

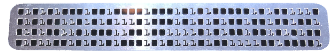
Neil Chandler

Database Chap. Knows Things.



Working in IT since 1988
Working with Oracle since about 1991
UKOUG Non-Executive Director

BLOG: http://chandlerdba.wordpress.com/
BlueSky: @chandlerdba

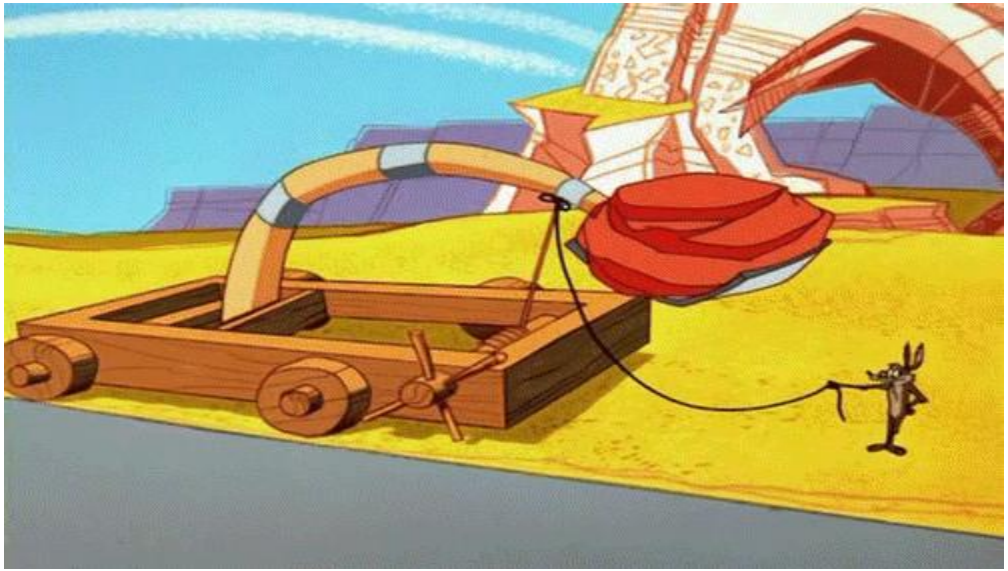


INTRODUCTION

What is your least favourite thing about Oracle

INTRODUCTION

DON'T INSULT THE OPTIMIZER WHEN YOU ARE SITTING NEXT TO THE OPTIMIZER LADY



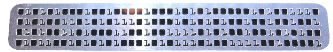
INTRODUCTION

DON'T INSULT THE OPTIMIZER WHEN YOU ARE SITTING NEXT TO THE OPTIMIZER LADY

AGENDA

- What are Histograms
- Why Chocolate is Evil
- Some Histogram Details

  Frequency / Top Frequency / Height Balanced / Hybrid
- A couple of Myths



OPTIMIZER

Poor Stats =>

Incorrect **Cardinality** Estimates =>

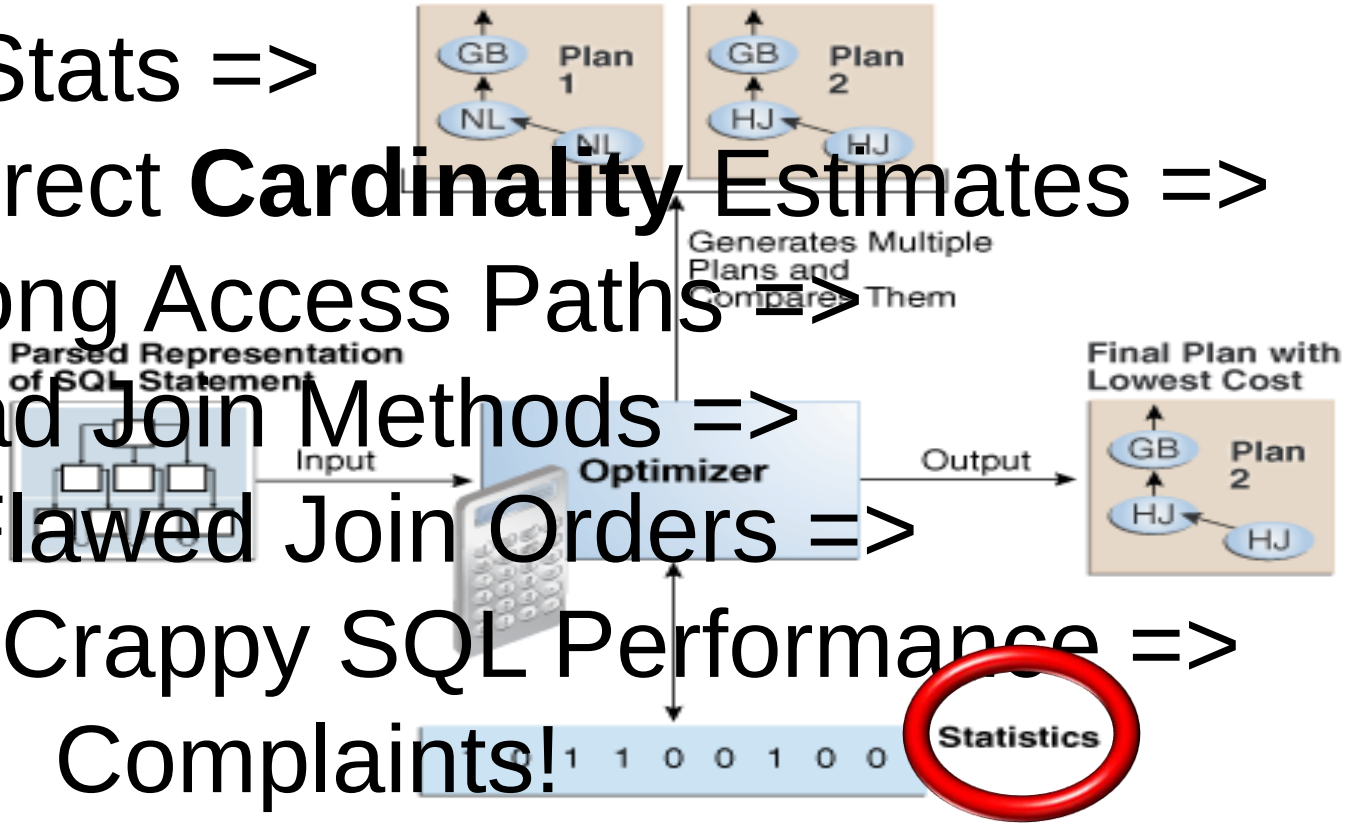
Wrong Access Paths =>

Bad Join Methods =>

Flawed Join Orders =>

Crappy SQL Performance =>

Complaints!



WHAT IS CARDINALITY?

The percentage of the table Oracle thinks you are going to return for a given predicate

Number of Rows in the table x DENSITY

DENSITY = 1 / Number-of-Distinct-Values-in-the-column

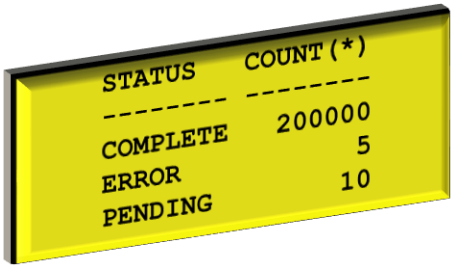
DBA_TAB_COL_STATISTICS.DENSITY
(or DBA_TAB_COLUMNS.DENSITY)

<u>Ignore this value if there is a Histogram on the column!</u>

HISTOGRAMS

- A type of *better* statistics
- Explains the data distributions in a column to improve **cardinality** estimates for a value
- But they may take a lot of effort to gather the extra stats
- May lead to a less efficient execution plan
- May give a less accurate description of your data
- Can promote plan instability

HISTOGRAMS



method_opt=>'FOR ALL COLUMNS SIZE 1'

DBA_TAB_COLUMNS – No Histogram

TABLE_NAME	COLUMN_NAM	DATA_TYPE	NUM_DISTINCT	DENSITY	HISTOGRAM	NUM_BUCKETS	LOW_DECODE	HIGH_DECODE
TEST_BASIC	STATUS	VARCHAR2	3	0.333333333333	NONE	1	COMPLETE	PENDING

DBA_TAB_HISTOGRAMS

TABLE_NAME	COLUMN_NAM	ENDPOINT_NUMBER	ENDPOINT_VALUE	ENDPOINT_DECODE (6 chars)
TEST_BASIC	STATUS			
TEST_BASIC	STATU			

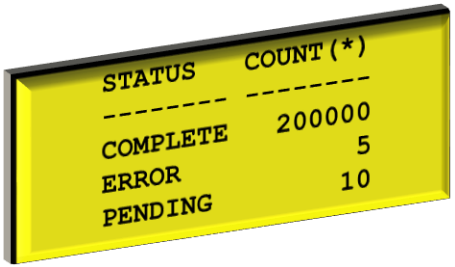
NO HISTOGRAM
Cardinality =
num_rows * (1 / NDV) =
num_rows * density =
200015 * (0.333) = 66671.6

DBMS_XPLAN – Execution

```
select ID from test_basic where status
```

Id	Operation	Name	Starts	E-Rows	E-Bytes	Cost (%CPU)	A-Rows	Buffers
0	SELECT STATEMENT		1			171 (100)	5	570
*1	TABLE ACCESS FULL	TEST_BASIC	1	66672	911K	171 (1)	5	570

HISTOGRAMS



method_opt=>'FOR ALL COLUMNS SIZE AUTO'

DBA_TAB_COLUMNS – Frequency Histogram

```
TABLE_NAME      COLUMN_NAM DATA_TYPE  NUM_DISTINCT         DENSITY HISTOGRAM  NUM_BUCKETS LOW_DECODE HIGH_DECODE
--------------- ---------- ---------- ------------ --------------- ---------- ----------- ---------- -----------
TEST_BASIC      STATUS     VARCHAR2              3 .333333333333333 FREQUENCY           3
```

200000-0 = 200000 rows with value of "COMPLETE"
200005-200000 = 5 rows with value of "ERROR"
200015-200005 = 10 rows with value of "PENDING"

DBA_TAB_HISTOGRAMS

```
TABLE_NAME      COLUMN_NAM ENDPOINT_NUMBER                          ENDPOINT_VALUE ENDPOINT_DECODE (6 chars)
--------------- ---------- --------------- ---------------------------------------- ------------------------
TEST_BASIC      STATUS              200000      3494923253000420000000000000000000000 COMPLE
TEST_BASIC      STATUS              200005      3599381620841760000000000000000000000 ERROQ
TEST_BASIC      STATUS              200015      4167894358757160000000000000000000000 PENDIN
```

DBMS_XPLAN – Execution Plan

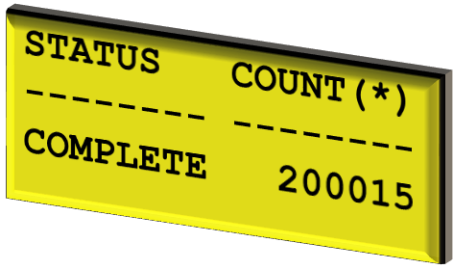
```
select ID from test_basic where status = 'ERROR';
-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                            | Name       | Starts | E-Rows |E-Bytes|Cost (%CPU)|A-Rows |Buffers|
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                     |            |      1 |        |       |    4 (100)|     5 |      6 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| TEST_BASIC |      1 |      5 |    70 |    4   (0)|     5 |      6 |
|*  2 |   INDEX RANGE SCAN                   | IDX_STATUS |      1 |      5 |       |    3   (0)|     5 |      4 |
-------------------------------------------------------------------------------------------------------------------
```

HISTOGRAMS : TIMING

The TIME OF DAY you gather stats is critical!

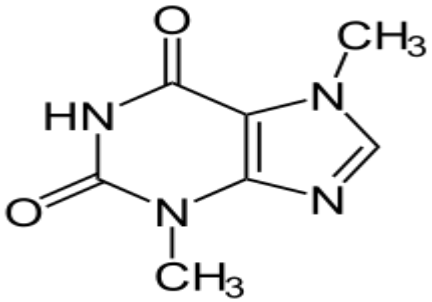
- 18:00 Processing Ends for the day
- 19:00 All PENDING records are processed and COMPLETE
- 20:00 All ERROR records are processed and COMPLETE
- 22:00 Gather Stats...

You only have status of COMPLETE



CHOCOLATE IS EVIL?

CHOCOLATE CAN BE EVIL!



Chocolate contains Theobromine, an "alkaloid" (like Cocaine and Caffeine)
A lethal dose of chocolate for a human is about 22lb / 10kilos

HISTOGRAMS ARE EVIL?

Oracle
LOVES
histograms

5-25% of columns will get Frequency histograms, but I have seen as high as 60%

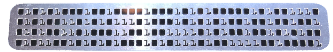
Seems to be higher % in 12. Height Balanced are replaced Hybrid (or Frequency or Top-Frequency)

Maybe 2-3% of columns get Hybrid/Height Balanced Histograms – but usually on the BIG tables…

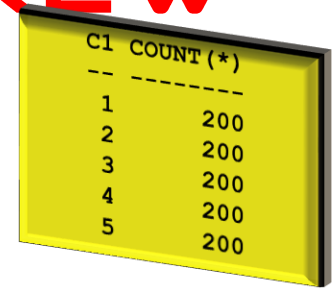
HISTOGRAM MYTH

HISTOGRAM MYTHS

You only get a histogram on skewed data



HISTOGRAM MYTHS : SKEW




```
create table test_uniform (c1 number not null) as
select mod(rownum,5)+1 from dba_objects where rownum < 1001;

Table Created
```

Need a query before size "auto" kicks in to create a histogram

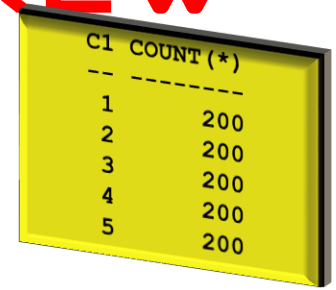
[We could use "skewonly" instead of auto to do this]

```
select count(*) from test_uniform where c1 > 4;
  COUNT(*)
----------
       200
```

```
SQL> select * from sys.col_usage$ where

TABLE_NAME   COL_NM EQUALITY EQUIJOIN NONEQUIJOIN RANGE LIKE NULL
------------ ------ -------- -------- ----------- ----- ---- ----
TEST_UNIFORM C1            0        0           0     1    0    0
```

(or you could use **DBMS_STATS.REPORT_COL_USAGE**)

HISTOGRAM MYTHS : SKEW




```
dbms_stats.gather_table_stats(null,'TEST_UNIFORM',
            method_opt=>'FOR ALL COLUMNS SIZE AUTO');
```

```
TABLE_NAME   COL_NM NUM_DISTINCT NUM_NULLS NUM_BUCKETS HISTOGRAM
------------ ------ ------------ --------- ----------- ---------
TEST_UNIFORM C1                5         0           5 FREQUENCY
```

USER_TAB_HISTOGRAMS

```
TABLE_NAME      COLUMN_NAM ENDPOINT_NUMBER ENDPOINT_VALUE
--------------- ---------- --------------- --------------
TEST_UNIFORM    C1                    200              1
TEST_UNIFORM    C1                    400              2
TEST_UNIFORM    C1                    600              3
TEST_UNIFORM    C1                    800              4
TEST_UNIFORM    C1                   1000              5
```

HISTOGRAM MYTHS : SKEW

Oracle
LOVES
histograms

5-25% of columns will get Frequency histograms, but I have seen as high as 60%

Seems to be higher % in 12. Height Balanced are replaced Hybrid (or Frequency or Top-Frequency)

Maybe 2-3% of columns get Hybrid/Height Balanced Histograms – but usually on the BIG tables…

Real Rule: Where you have a predicate, you're probably getting a histogram…

(except for equality predicates against unique not null columns)

HISTOGRAMS ARE EVIL?

Where you have a predicate,
you're probably getting a histogram…

- Histograms Consume Resources
  to create and maintain
  (esp. Hybrid and Height Balanced)
- Histograms Consume Resources to Use - they are resident in the dictionary cache
- Histograms Consume Resources to Store
- Histograms *may* make your plans worse
- Histograms *may* make your plans less stable

HISTOGRAMS TYPES

Pre-12C

Frequency

Where there are fewer distinct values than entries/buckets
(up to 254 entries/buckets)

Height Balanced

Where there are more distinct values than entries/buckets

New for 12C

Top-Frequency

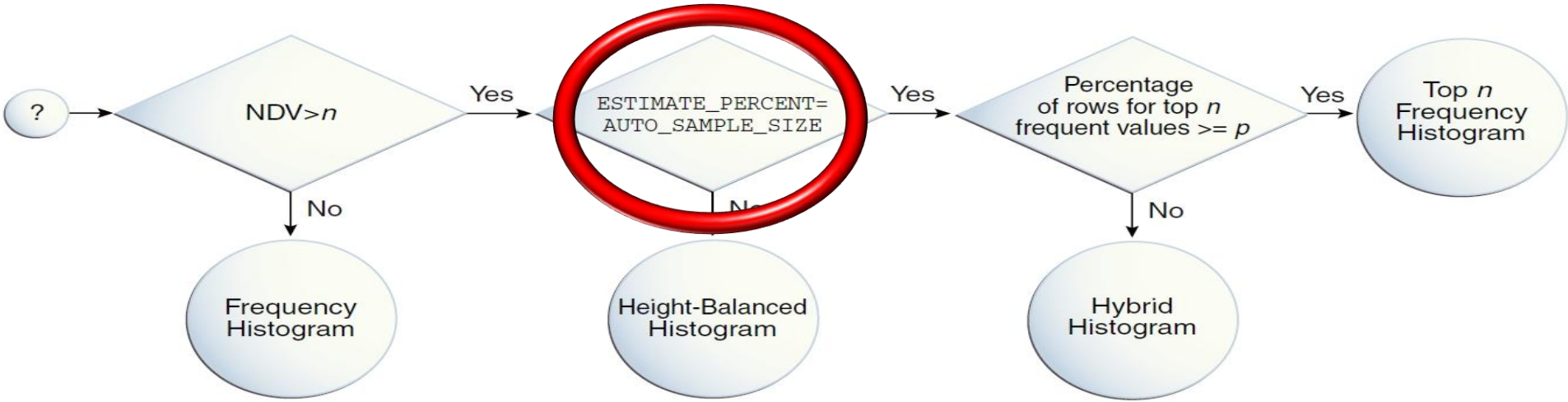
Where there are slightly more "irrelevant" distinct values than entries

Hybrid

Replaces Height Balanced

Allowed to have 2048 entries/buckets (default still 254)

HISTOGRAMS TYPE DESCISION

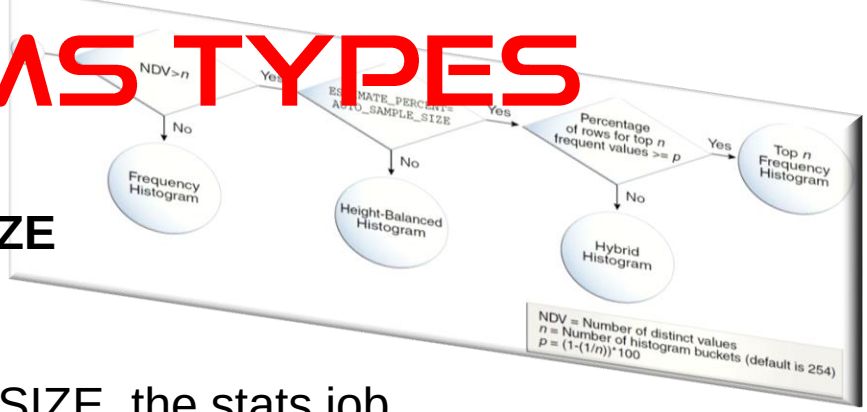


NDV = Number of distinct values
n = Number of histogram buckets (default is 254)
p = (1-(1/*n*))*100

Diagram stolen from Oracle SQL Tuning guide

HISTOGRAMS TYPES

You only get the 2 new types of Histogram
if **ESTIMATE_PERCENT=AUTO_SAMPLE_SIZE**



From 11G, if left to default to AUTO_SAMPLE_SIZE, the stats job performs FULL TABLE SCAN instead of Adaptive Sampling

Histogram Stats Processing evolved between 10G and 12C:

10G - sample data for every column individually, and re-samples larger based upon perceived NDV correctness and number of NULLS it finds

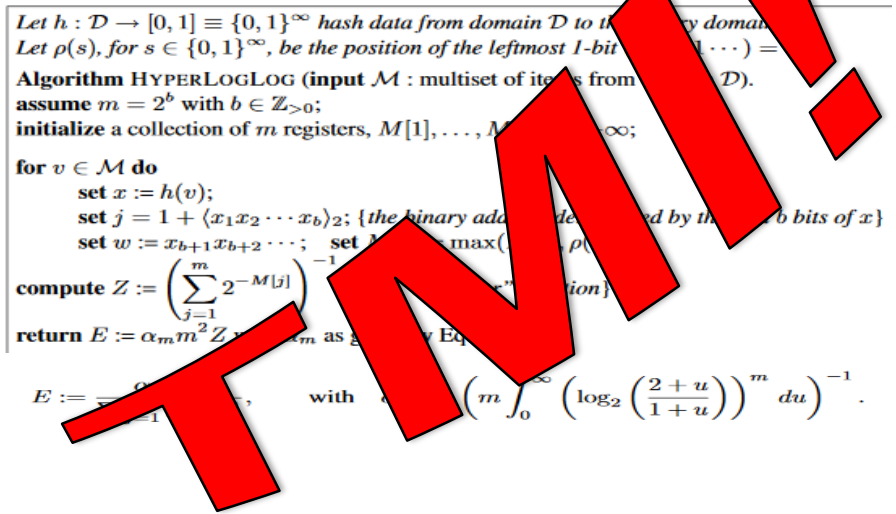
11G - typically one* sample for all columns with histograms

12C - Frequency-type Histograms gathered in a single pass using new **APPROXIMATE_NDV** processing

*if some column(s) have lots of NULLs, we may get a multiple samples

APPROXIMATE_NDV

12C is using ***awesome*** maths using "HyperLogLog" algorithm for near perfect cardinality calculations in a single table scan



TMI!!

DBMS_STATS.APPROXIMATE_NDV_ALGORITHM

"ADAPTIVE SAMPLING" / "HYPERLOGLOG" / "REPEAT OR HYPERLOGLOG"

APPROXIMATE_NDV

TABLE SCAN:

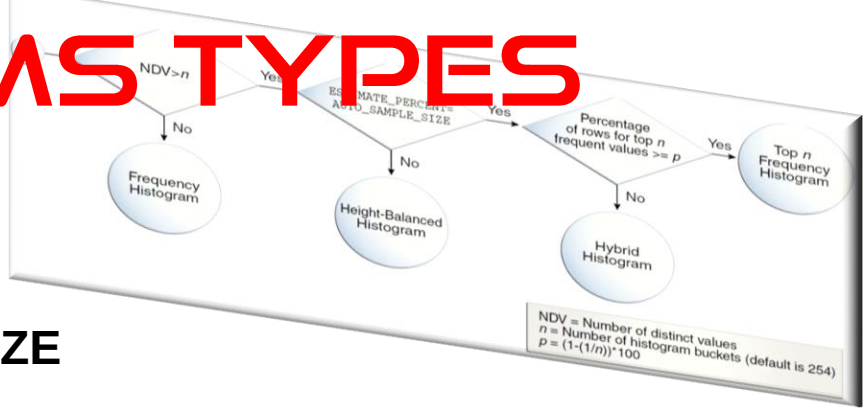
Oracle basically hashes every new value a[illegible]stores the hash in a "column synop[illegible]s"

When the hash table (16[illegible]lues) is full
it throws ½ of the lower h[illegible]value[illegible]av[illegible] "do[illegible]in splitting"
&[illegible]

If it fills [illegible]it throws [illegible]lf of the values away again
whe[illegible]he 1st bit is 0 and keeps going

TMI!!

HISTOGRAMS TYPES

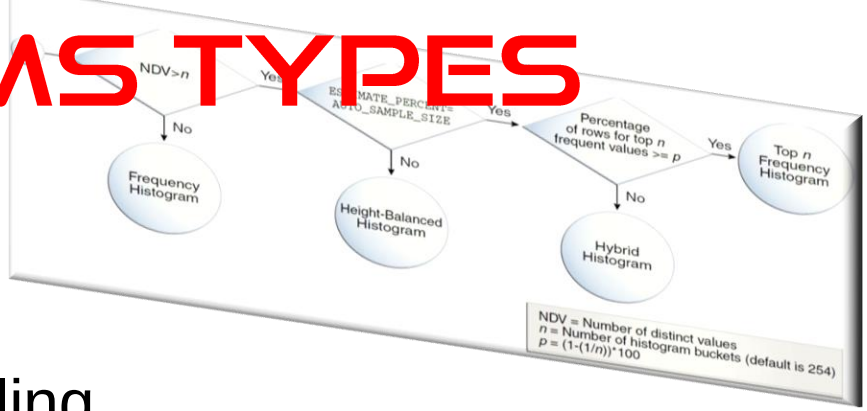


If ESTIMATE_PERCENT=AUTO_SAMPLE_SIZE

gathering Frequency and Top-Frequency Histograms is free*
and very accurate

(*tiny bit of CPU)

HISTOGRAMS TYPES



Adaptive Sampling

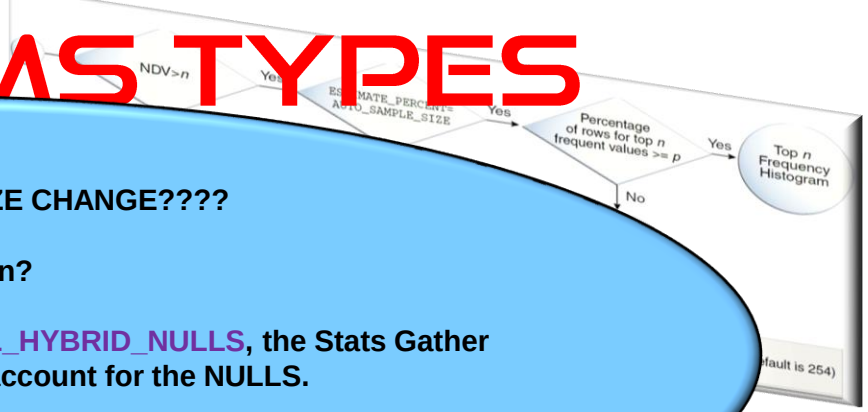
Height Balanced and **Hybrid**
histograms still use Adaptive Sampling
(as do Frequency Histograms pre-12C)

```
method_opt=>'FOR ALL COLUMNS SIZE AUTO FOR COLUMNS SIZE 12 COL_HYBRID'
```

(DBA_TAB_COLUMNS)

COLUMN_NAME	NUM_DISTINCT	NULLS	NUM_BUCKETS	HISTOGRAM	SAMPLE_SIZE
TST_ID	53335	0	1	NONE	53335
COL_HYBRID	**29**	**0**	**12**	**HYBRID**	**5590**
COL_FREQUENCY	10	1	10	FREQUENCY	53334
COL_TEXT	46520	1	1	NONE	53334

HISTOGRAMS TYPES



Adapti

Ad
wh

metho

WHY DOES THE SAMPLE SIZE CHANGE????

What's going on?

To get a sample size of about 5,500 rows in COL_HYBRID_NULLS, the Stats Gather needed to *sample more* blocks to account for the NULLS.

There is only 1 sample, so COL_HYBRID got a bigger sample of 9,276 rows

LS'

COLUMN_NAME	NUM_DISTINCT	NULLS	NUM_BUCKETS	HISTOGRAM	SAMPLE_SIZE
TST_ID	53335	0	1	NONE	53335
COL_HYBRID	**29**	**0**	**12**	**HYBRID**	**9276**
COL_HYBRID_NULLS	**19**	**21671**	**12**	**HYBRID**	**5503**
COL_FREQUENCY	10	1	10	FREQUENCY	53334
COL_TEXT	46912	1	1	NONE	53334

HISTOGRAMS TYPES

SQL for stats gathering is FTS

```
SELECT /*+ lots-of-hints */
  TO_CHAR(COUNT("TST_ID")),
  substrb(dump(MIN("TST_ID"),16,0,64),1,240),
  substrb(dump(MAX("TST_ID"),16,0,64),1,240),
  TO_CHAR(COUNT("COL_HYBRID")),
  substrb(dump(MIN("COL_HYBRID"),16,0,64),1,240),
  substrb(dump(MAX("COL_HYBRID"),16,0,64),1,240),
  TO_CHAR(COUNT("COL_HYBRID_NULLS")),
  substrb(dump(MIN("COL_HYBRID_NULLS"),16, 0,64),1,240),
  substrb(dump(MAX("COL_HYBRID_NULLS"),16,0,64),1,240),
  TO_CHAR(COUNT("COL_FREQUENCY")),
  substrb(dump(MIN("COL_FREQUENCY"),16,0,64), 1,240),
  substrb(dump(MAX("COL_FREQUENCY"),16,0,64),1,240),
  TO_CHAR(COUNT("COL_TEXT")),
  substrb(dump(MIN("COL_TEXT"),16,0,64),1,240),
  substrb(dump(MAX("COL_TEXT"),16,0,64),1,240),
  COUNT(rowidtochar(rowid))
FROM "NEIL"."TEST_SAMPLING" t
  /* ACL,NIL,NIL,TOPN,NIL,NIL,TOPN,NIL,NIL,TOPN,NIL,
  NIL,NDV,NIL,NIL,RWID,U254,U12,U12,U254,U254U */

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT AGGREGATE (cr=248 pr=0 pw=0 time=107302 us starts=1)
     53335      53335      53335   OPTIMIZER STATISTICS GATHERING  (cr=248 pr=0 pw=0 time=141938 us starts=1 cost=68 size=1013365 card=53335)
     53335      53335      53335    TABLE ACCESS FULL TEST_SAMPLING (cr=248 pr=0 pw=0 time=22027 us starts=1 cost=68 size=1013365 card=53335)
```


HISTOGRAMS TYPES

Some additional information for the Approximate NDV calculations

Pulls 10 rows out of the DB for the HyperLogLog Calc

```
SELECT /*+ lots-of-hints */
  substrb(dump("COL_FREQUENCY",16,0,64),1,240) val,
  rowidtochar(rowid) rwid
FROM "NEIL"."TEST_SAMPLING" t
WHERE rowid IN
(chartorowid('AAASLXAAOAAAACDAAA'),chartorowid('AAASLXAAOAAAACDAAB'),chartorowid('AAASLXAAOAAAACDAAC'), chartorowid('AAASLXAAOAAAACDAAD'),
 chartorowid('AAASLXAAOAAAACDAAE'),chartorowid('AAASLXAAOAAAACDAAF'),chartorowid('AAASLXAAOAAAACDAAG'),
 chartorowid('AAASLXAAOAAAACDAAI'),chartorowid('AAASLXAAOAAAACDAAJ'),chartorowid('AAASLXAAOAAAACDAAR')) ORDER BY "COL_FREQUENCY"

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
        10         10         10  SORT ORDER BY (cr=1 pr=0 pw=0 time=43 us starts=1 cost=2 size=19 card=1)
        10         10         10   INLIST ITERATOR  (cr=1 pr=0 pw=0 time=20 us starts=1)
        10         10         10    TABLE ACCESS BY USER ROWID TEST_SAMPLING (cr=1 pr=0 pw=0 time=10 us starts=10 cost=1 size=19 card=1)
```


HISTOGRAMS TYPES

Create GTT to hold sample data

```
CREATE global TEMPORARY TABLE sys.ora_temp_1_ds_560022 sharing=none
ON COMMIT preserve rows cache noparallel
AS
  SELECT
    /*+ lots-of-hints */
    "COL_HYBRID",
    "COL_HYBRID_NULLS",
    rowid SYS_DS_ALIAS_0
  FROM "NEIL"."TEST_SAMPLING" sample ( 17.3698837797) t
  WHERE 1 = 2

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         0          0          0  LOAD AS SELECT  ORA_TEMP_1_DS_560022 (cr=0 pr=0 pw=0 time=30 us starts=1)
         0          0          0   FILTER  (cr=0 pr=0 pw=0 time=2 us starts=1)
         0          0          0    TABLE ACCESS SAMPLE TEST_SAMPLING (cr=0 pr=0 pw=0 time=0 us starts=0 cost=68 size=176016 card=9264)
```


HISTOGRAMS TYPES

Grab the sample data - the Gather knows there's NULLs from the earlier FTS

```
INSERT /*+ append */
INTO sys.ora_temp_1_ds_560022
SELECT
  /*+ lots-of-hints */
  "COL_HYBRID",
  "COL_HYBRID_NULLS",
  rowid SYS_DS_ALIAS_0
FROM "NEIL"."TEST_SAMPLING" sample ( 17.3698837797) t
UNION ALL
SELECT "COL_HYBRID",
  "COL_HYBRID_NULLS",
  SYS_DS_ALIAS_0
FROM sys.ora_temp_1_ds_560022
WHERE 1 = 0

Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         0          0          0  LOAD AS SELECT  ORA_TEMP_1_DS_560022 (cr=248 pr=0 pw=28 time=51969 us starts=1)
      9276       9276       9276   UNION-ALL  (cr=248 pr=0 pw=0 time=6110 us starts=1)
      9276       9276       9276    TABLE ACCESS SAMPLE TEST_SAMPLING (cr=248 pr=0 pw=0 time=3440 us starts=1 cost=68 size=176016 card=9264)
         0          0          0    FILTER  (cr=0 pr=0 pw=0 time=1 us starts=1)
         0          0          0     TABLE ACCESS FULL ORA_TEMP_1_DS_560022 (cr=0 pr=0 pw=0 time=0 us starts=0 cost=29 size=310384 card=8168)
```

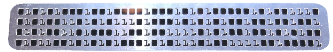

HISTOGRAMS TYPES

and some final bits of index information

```
SELECT
  /*+  hints */
  COUNT(*)                                          AS nrw,
  COUNT(DISTINCT sys_op_lbid(74456,'L',t.rowid)) AS nlb,
  NULL                                              AS ndk,
  sys_op_countchg(substrb(t.rowid,1,15),1)       AS clf
FROM "NEIL"."TEST_SAMPLING" t
WHERE "TST_ID" IS NOT NULL



Rows (1st) Rows (avg) Rows (max)  Row Source Operation
---------- ---------- ----------  ---------------------------------------------------
         1          1          1  SORT GROUP BY (cr=107 pr=0 pw=0 time=35039 us starts=1)
     53335      53335      53335   INDEX FULL SCAN TEST_SAMPLING_PK
                                       (cr=107 pr=0 pw=0 time=7045 us starts=1 cost=107 size=640020 card=53335)(object id 74456)
```


SAMPLING THREAT

if you are SAMPLING data
(Hybrid / Height Balanced)
rare values will appear
and disappear, potentially
causing plan stability issues

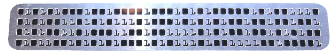


HISTOGRAM MYTHS

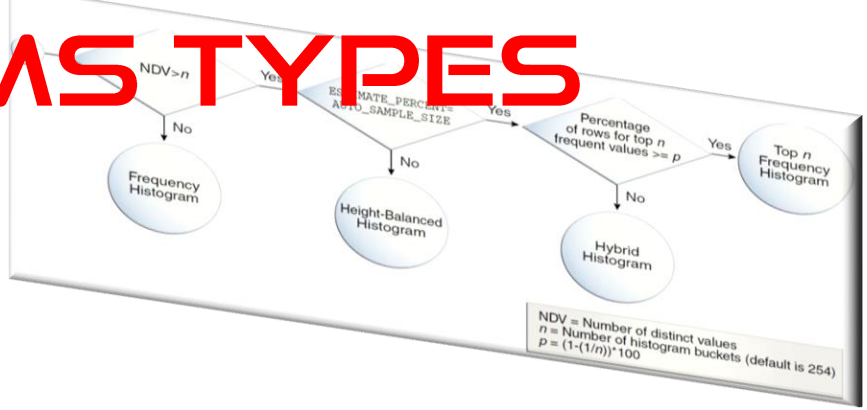
Histograms are only for indexed columns?

They help with Join Cardinality and therefore Join Methods too

If the optimizer *can* use stats to help,
it *will* be using stats to help



HISTOGRAMS TYPES



Frequency

We have more "buckets" than NDV's

Top-Frequency

A few more NDV's than entries (buckets) available but throw away "insignificant" values which rarely occur
Like ALL histograms, it **must** record Low/High Values*

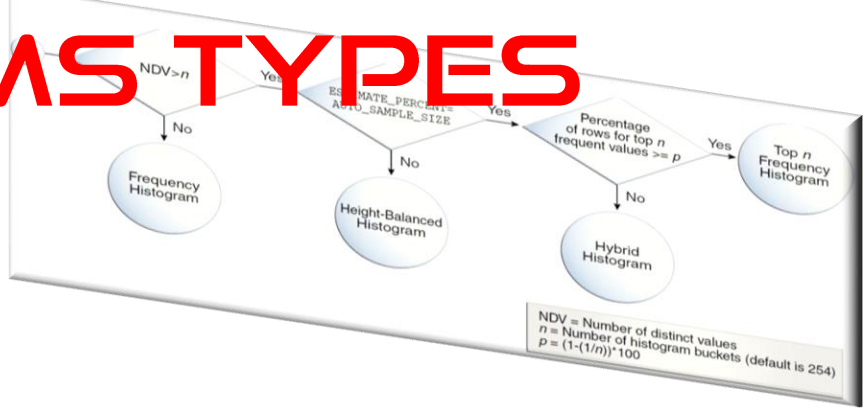
n=254
p=(1-(1/254))*100=
> 99.6% of the data must fit into 254 entries. 0.4% is "insignificant"

n=20
p=(1-(1/20))*100
> 95% of the data must fit into 20 entries. 5% is "insignificant"

*some bugs in 12.1 where it gets the low/high values wrong! Changed operation from 12.2

HISTOGRAMS TYPES

Height-Balanced & Hybrid



Looking to identify and record "**popular**" values.
Must record lowest and highest values

Example: **21** Distinct Values in this *sample*
rows=**60**
size (buckets)=**12**
('FOR ALL COLUMNS SIZE 12')

Oracle must sort the data set from the sample (*this is an expensive operation*)

We have 60 rows and 12 buckets. The Histogram will be build by capturing every (60/12) 5th value

TEAM	COUNT(*)	TEAM	COUNT(*)
Ajax	1	Juventus	4
Arsenal	3	Legia Warsaw	1
Athletic Bilbao	1	Liverpool	4
Athletico Madrid	1	Man City	3
Barcelona	6	Man Utd	9
BayernM	11	Maribor	1
Borussia Dortmund	2	Paris Saint Germain	1
CSKA Sofia	4	Porto	1
Dinamo Zagreb	1	Real Madrid	1
Hertha Berlin	1	Sunderland	1
Internazionale	3		

HISTOGRAMS TYPES

Height-Balanced: lowest, highest, and every 5th value

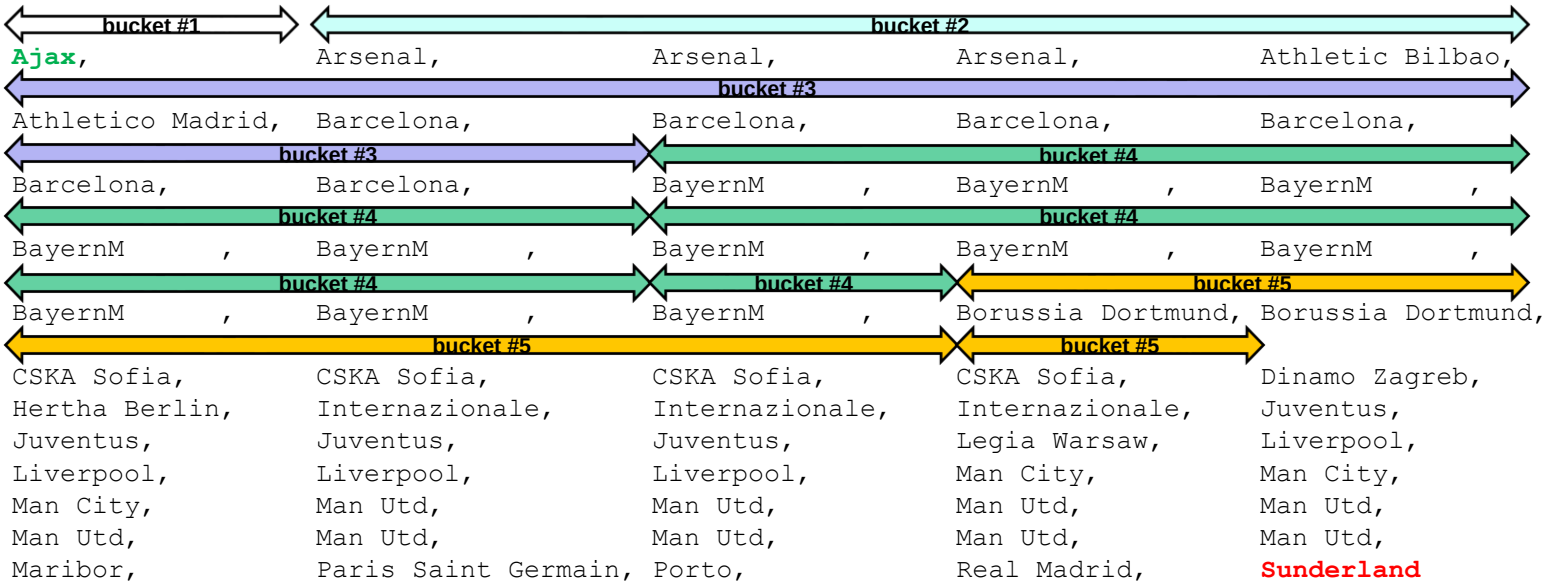
```
 0 Ajax
 1 Ajax,                Arsenal,              Arsenal,              Arsenal,              Athletic Bilbao,
 2 Athletico Madrid,    Barcelona,            Barcelona,            Barcelona,            Barcelona,
 3 Barcelona,
 4 BayernM,
 5 BayernM,
 6 CSKA Sofia,
 7 Hertha Berlin,
 8 Juventus,
 9 Liverpool,
10 Man City,
11 Man Utd,
12 Maribor,

COLUMN_NAM ENDPOINT_NU
---------- ----------
TEAM
TEAM
TEAM
TEAM                                                                          needed)
TEAM                5
TEAM                6      355214584638972000000000000000000000 Dinamo Zagreb
TEAM                7      386612389769899000000000000000000000 Juventus
TEAM                8      396753594576542000000000000000000000 Liverpool
TEAM                9      401782997701836000000000000000000000 Man City
TEAM               11      401782997701836000000000000000000000 Man Utd          <= popular
TEAM               12      433342427351037000000000000000000000 Sunderland
```

TEAM	COUNT(*)	TEAM	COUNT(*)
Ajax	1	Juventus	4
Arsenal	3	Legia Warsaw	1
Athletic Bilbao	1	Liverpool	4
Athletico Madrid	1	Man City	3
Barcelona	6	Man Utd	9
BayernM	11	Maribor	1
Borussia Dortmund	2	Paris Saint Germain	1
CSKA Sofia	4	Porto	1
Dinamo Zagreb	1	Real Madrid	1
Hertha Berlin	1	Sunderland	1
Internazionale	3		

HISTOGRAMS TYPES

Hybrid: lowest, highest and every 5th value BUT

- *no value can occupy more than 1 bucket*
- *duplicate values are moved into the same bucket*
- *buckets can vary in size*



HISTOGRAMS TYPES

Hybrid:

```
   (bucket size)
 1 ( 1) Ajax
 5 ( 4) Arsenal, Arsenal, Arsenal, Athletic Bilbao,
12 ( 7) Athletico Madrid, Barcelona, Barcelona, Barcelona, Barcelona, Barcelona, Barcelona,
23 (11) BayernM, BayernM, BayernM, BayernM, BayernM, BayernM, BayernM, BayernM, BayernM, BayernM, BayernM,
29 ( 6) Borussia Dortmund, Borussia Dortmund, CSKA Sofia, CSKA Sofia ,CSKA Sofia, CSKA Sofia,
34 ( 5) Dinamo Zagreb, Hertha Berlin, Internazionale, Internazionale, Internazionale,
38 ( 4) Juventus, Juventus, Juventus, Juventus,
43 ( 5) Legia Warsaw,  Liverpool, Liverpool, Liverpool, Liverpool,
55 (12) Man City, Man City, Man City, Man Utd, Man Utd, Man Utd, Man Utd, Man Utd, Man Utd, Man Utd, Man Utd, Man Utd
57 ( 2) Maribor,  Paris Saint Germain
58 ( 2) Porto,
60 ( 1) Real Madrid, Sunderland
```

COLUMN_NAM	ENDPOINT	ENDPOINT_VALUE	DECODE	ENDPOINT_REPEAT_COUNT	
TEAM	1	339656953492716000000000000000000000	Ajax	1	
TEAM	5	339860309322837000000000000000000000	Athlet	1	
TEAM	12	344668049167009000000000000000000000	Barcel	6	<= Popular (> orig BktSize 5)
TEAM	23	34	BayernM	11	<= Popular
TEAM	29	35	CSKA So	4	
TEAM	34	38	Intern	3	
TEAM	38	38	Juvent	4	
TEAM	43	39	Liverp	4	
TEAM	55	401782997701836000000000000000000000	Man Ut	9	<= Popular
TEAM	57	417360207059972000000000000000000000	Paris	1	
TEAM	58	417644164193770000000000000000000000	Portn	1	
TEAM	60	433342427351037000000000000000000000	Sunder	1	

This is actually a sampled Frequency Histogram on Popular Values

HISTOGRAMS TYPES

Hybrid and Height Balanced Histograms

It's all about being *POPULAR*!

CARDINALITY CALCS

Cardinality Calculation (12.2C) are undocumented,
a little complex & different for each Histogram type

They are subject to change with no notice

Frequency

If the Predicate Value is in the Histogram = Num Rows** stored in the Histogram
If the Predicate Value is NOT in the Histogram = Least Popular Value from Histogram / 2

Top Frequency

Value in Histogram = Num Rows** stored in the Histogram
Value not in Histogram = Least Popular Value from Histogram
(the "/2" is NOT applied in 12.2)

***Actually: Num Rows In The Histogram* x *(Num Rows in Table / Sample Size)*

CARDINALITY CALCS

Height-Balanced and Hybrid are all about Popular Values:

Height Balanced

NewDensity= ((Bucket# - PopularBucket#) / Bucket#) / (NDV/PopValueCnt)

Popular = Num Rows * (Num Buckets for Value / Total Num Buckets)
Non-Popular with Endpoint = Num Rows * *NewDensity*
Non-Popular without Endpoint = Num Rows * *NewDensity / 2 (not sure about the "/ 2" in 12.2)*

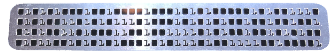
NOTE: Because is takes 2 buckets to identify a Popular value, your precision is at best ½ the number of buckets

Hybrid

Popular = ENDPOINT_REPEAT_COUNT * (NUM_ROWS / SAMPLE_SIZE)
Non-Popular with Endpoint = Num Rows * the largest of
[*NewDensity* or
ENDPOINT_REPEAT_COUNT/SAMPLE_SIZE]
Non-Popular without Endpoint = Num Rows * *NewDensity*

HISTOGRAM MYTHS

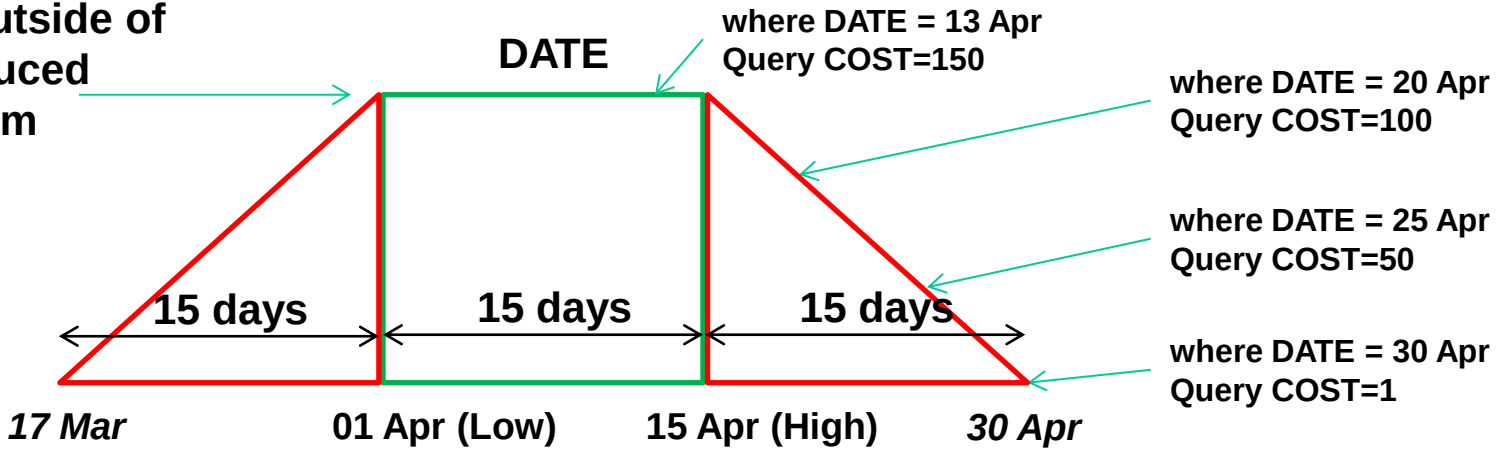
Your plan is always better with a histogram



HISTOGRAM MYTHS

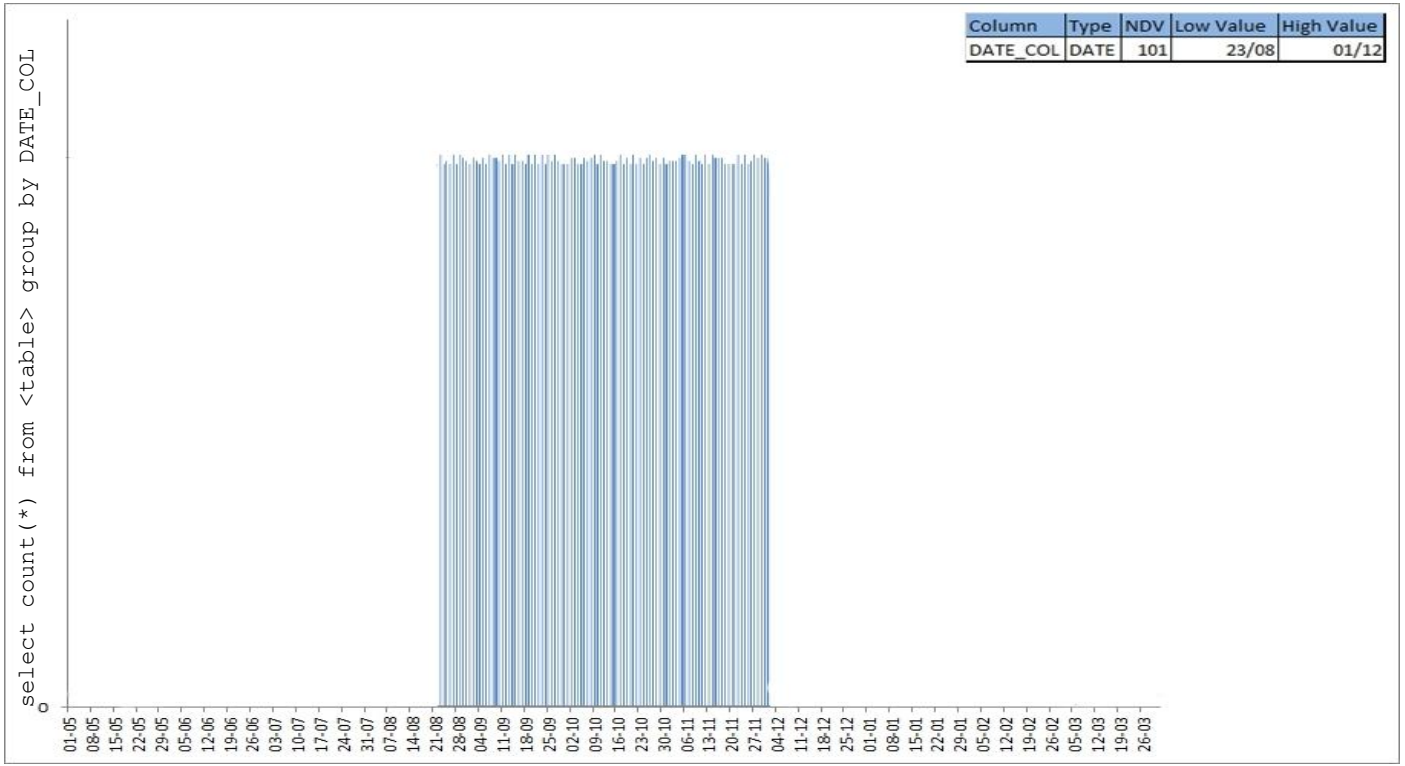
The low/high value threat

Query COST outside of 01-15 April reduced by *distance* from high/low range



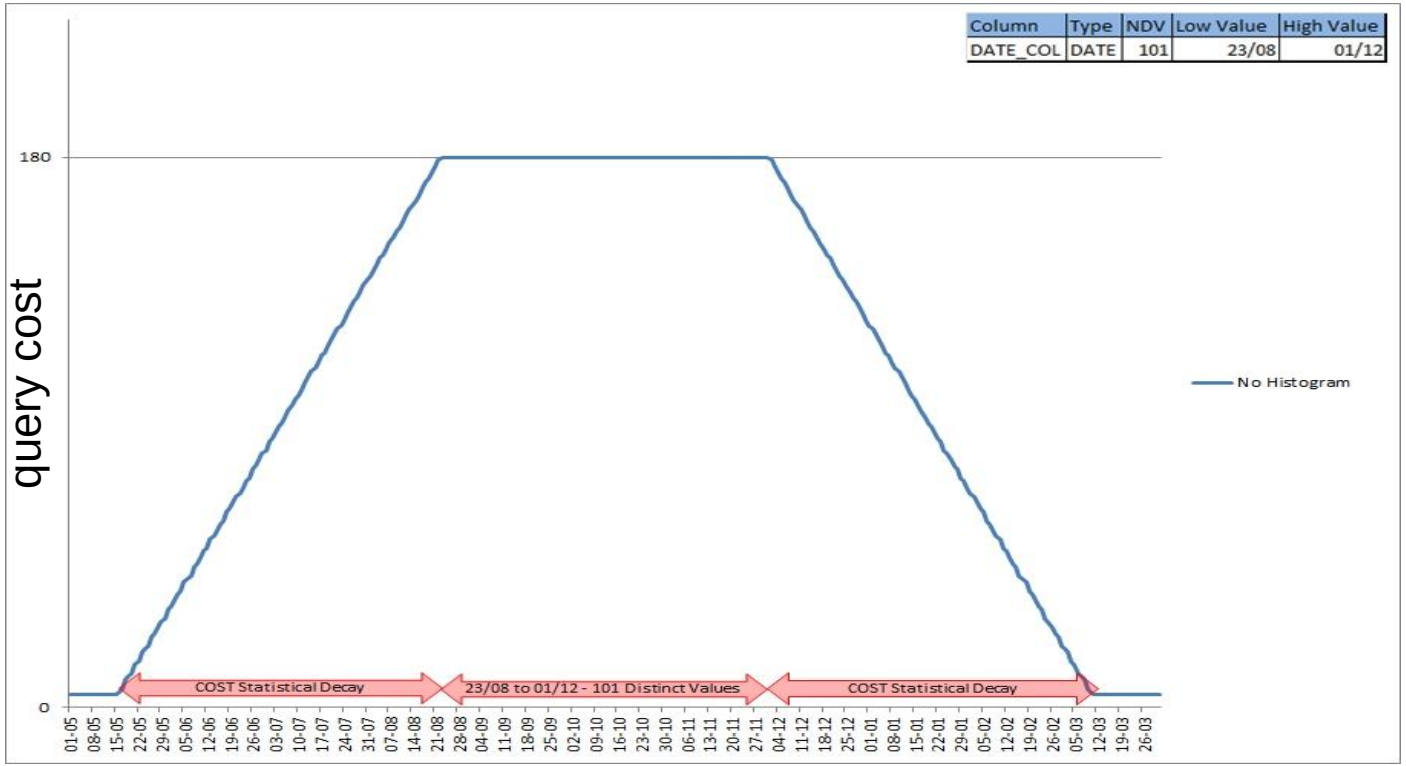
HISTOGRAM MYTHS

Well, here's a potential threat...



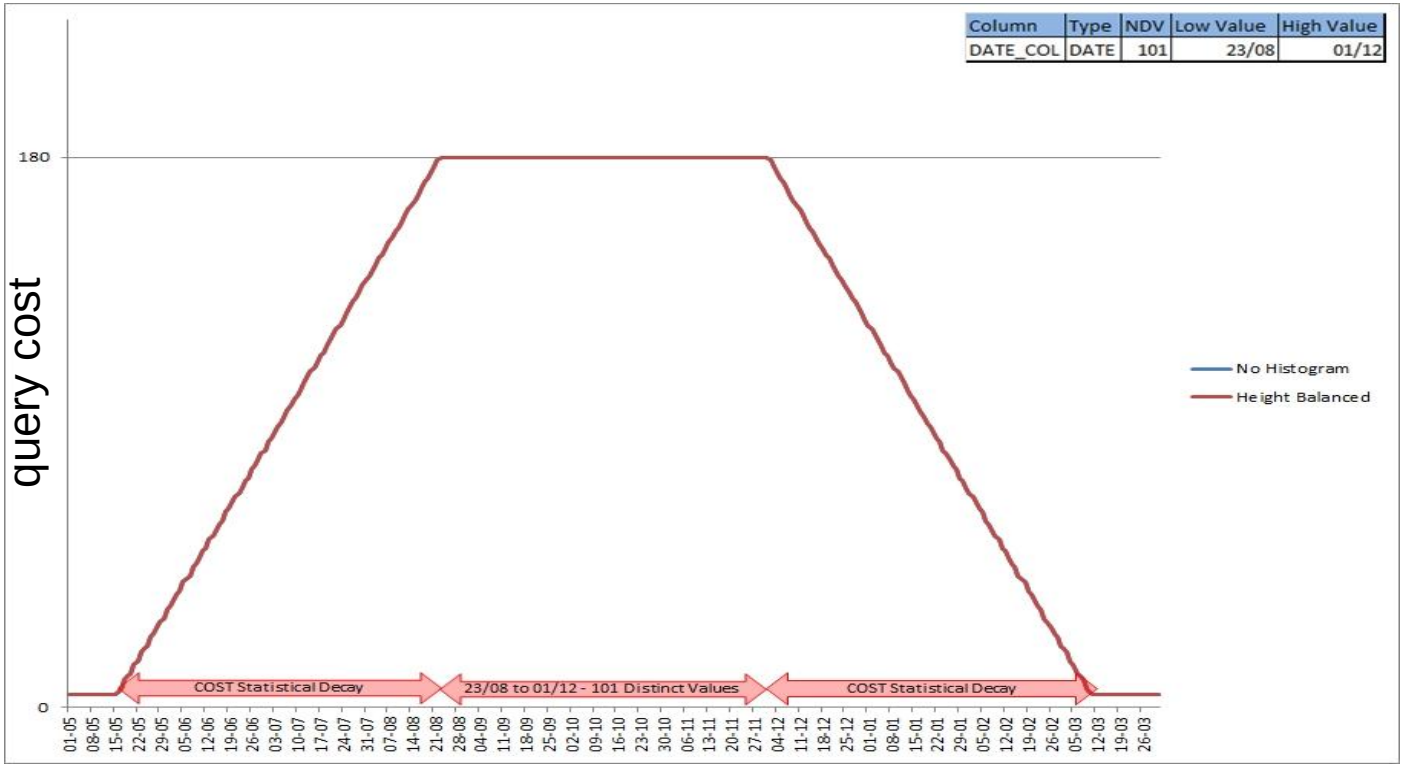
HISTOGRAM MYTHS

Well, here's a potential threat... Statistical Decay



HISTOGRAM MYTHS

Well, here's a potential threat...

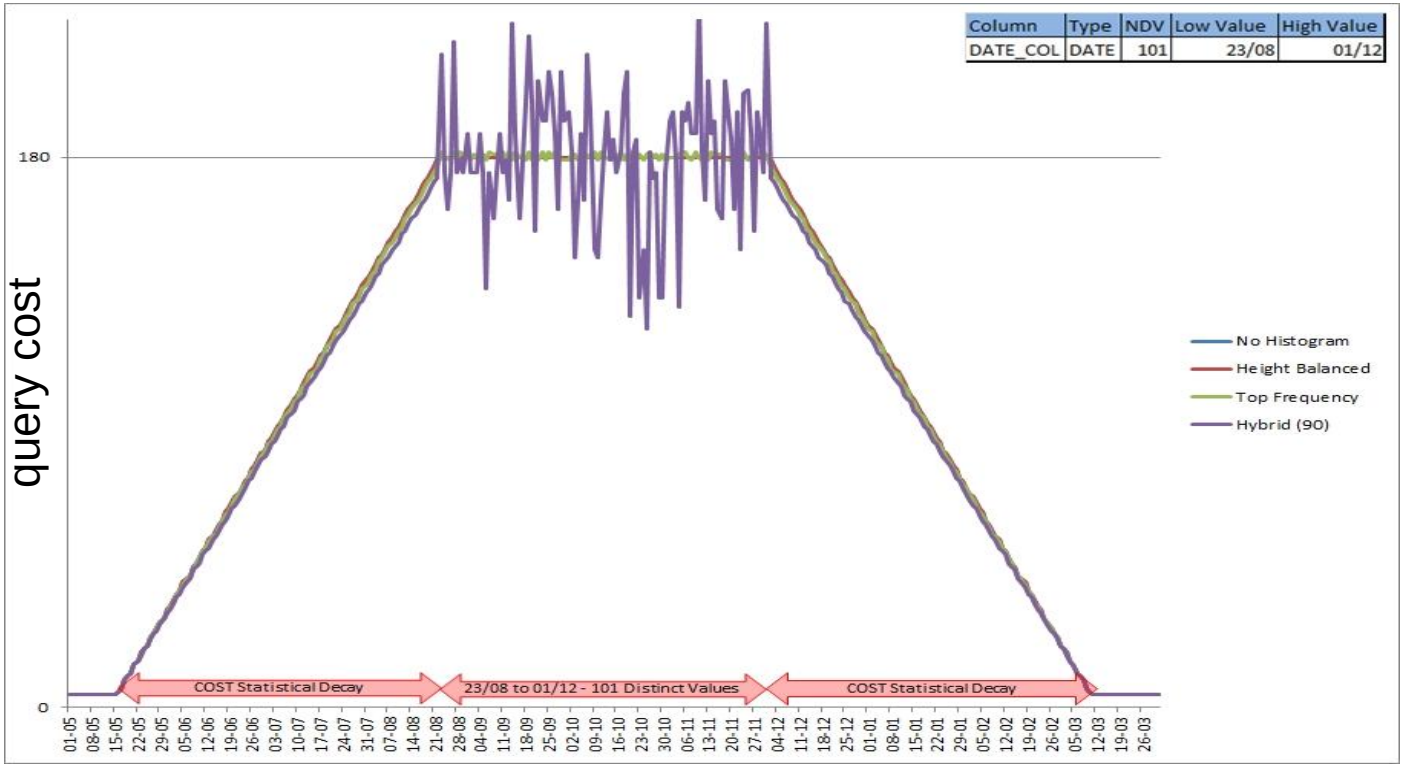


HISTOGRAM MYTHS

Well, here's a potential threat...

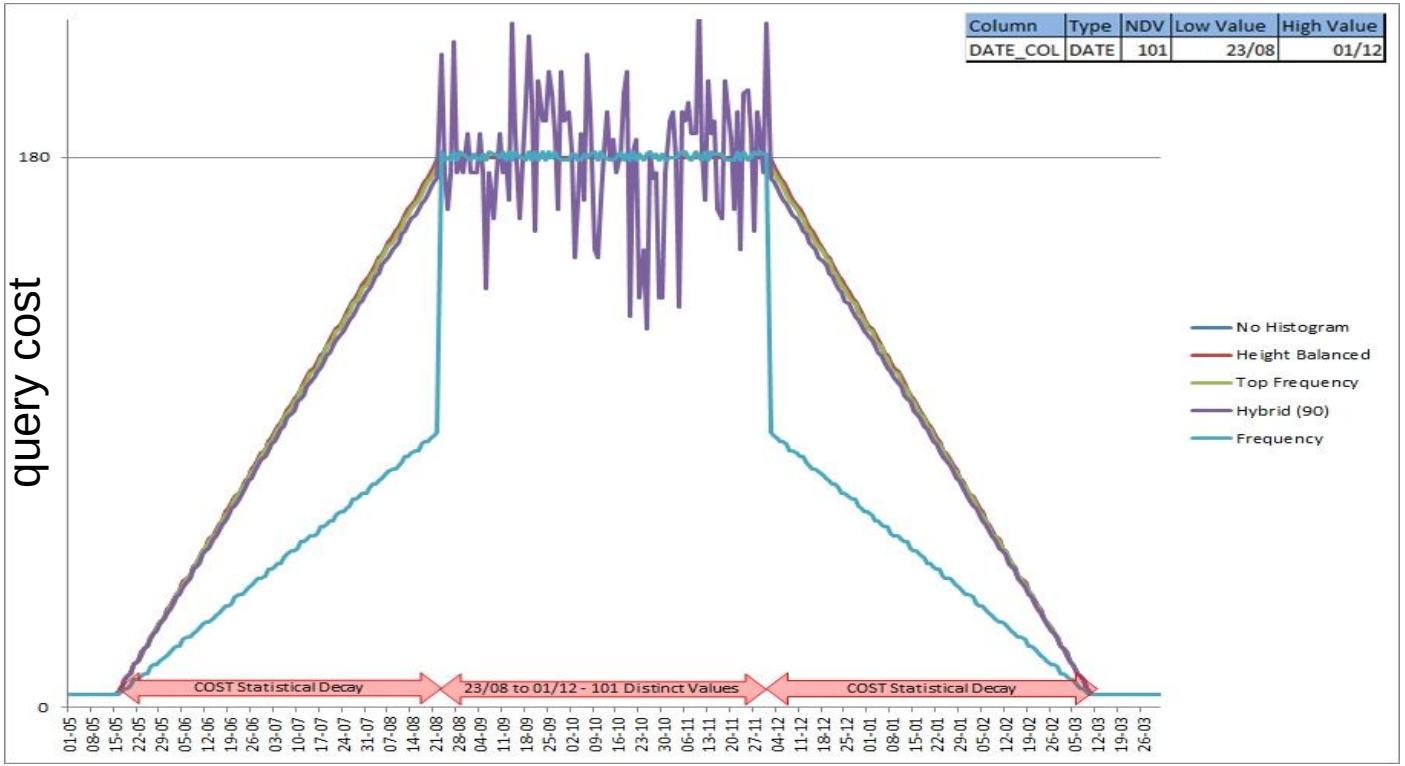
HISTOGRAM MYTHS

Well, here's a potential threat...



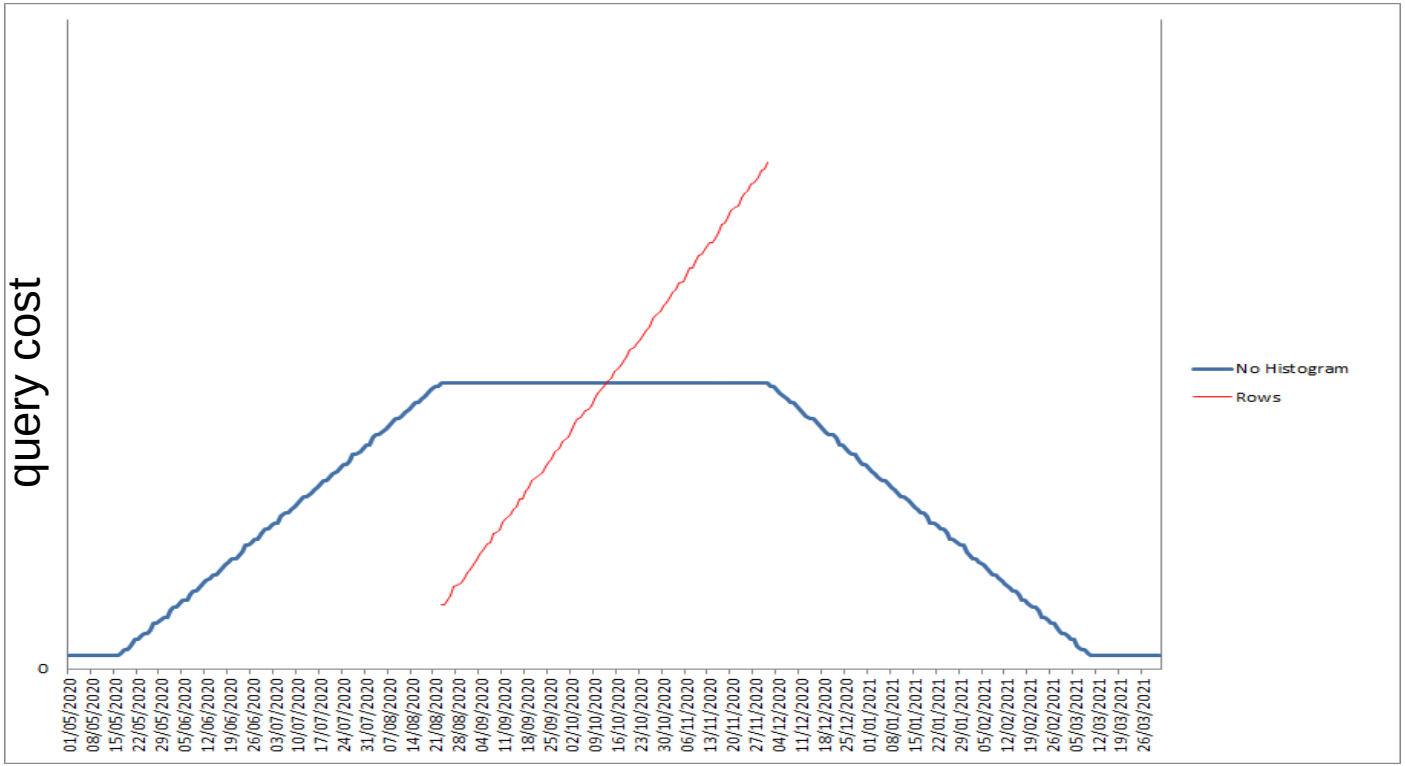
HISTOGRAM MYTHS

Well, here's a potential threat...



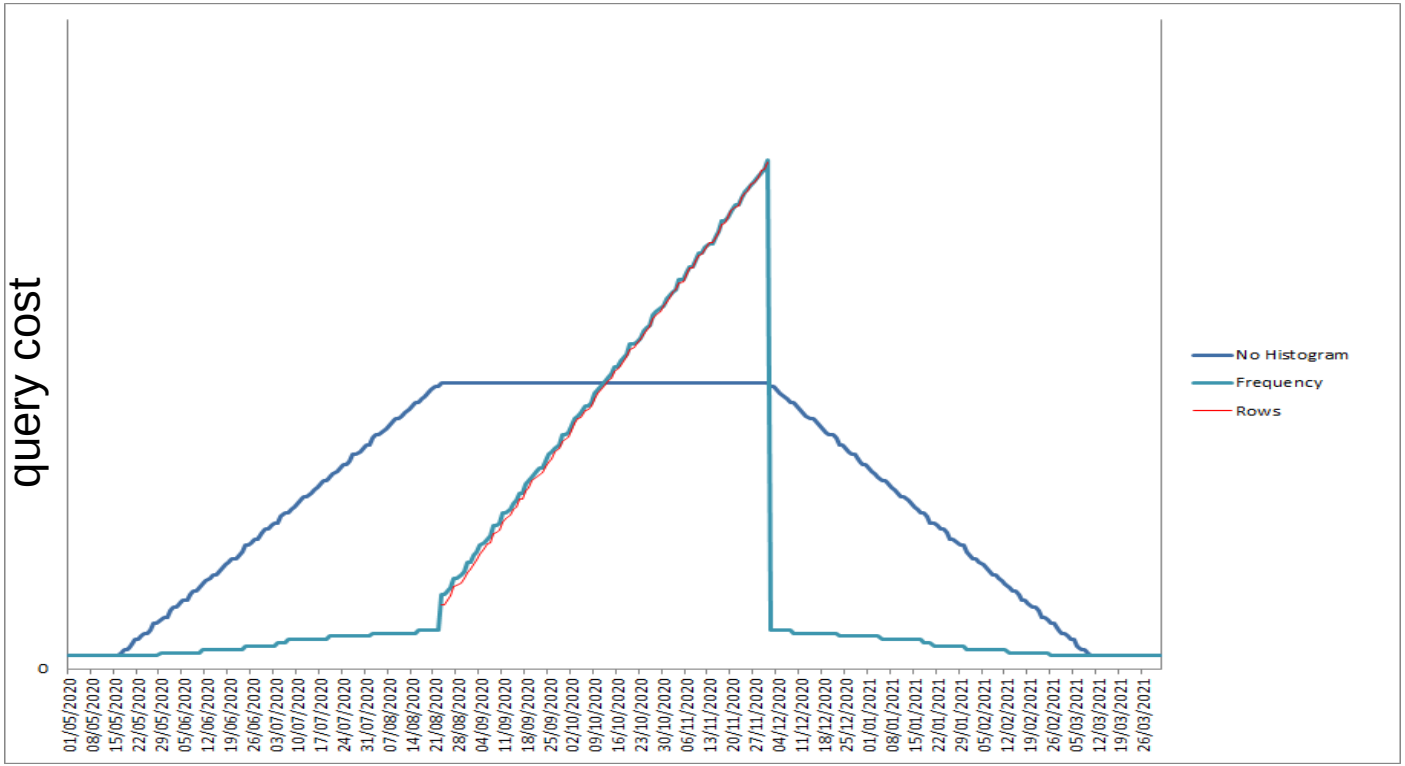
HISTOGRAM MYTHS

But with a skewed data set



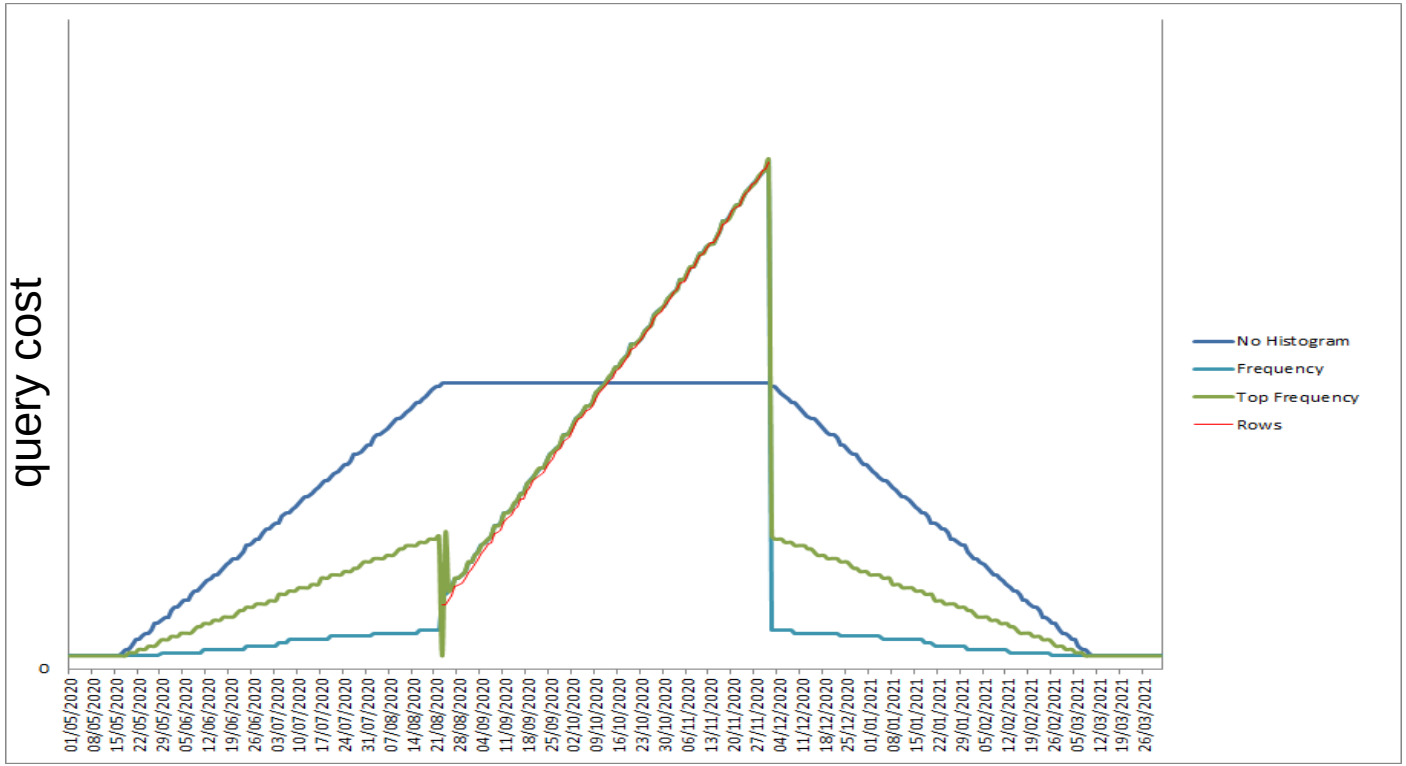
HISTOGRAM MYTHS

Frequency is awesome, but with threats



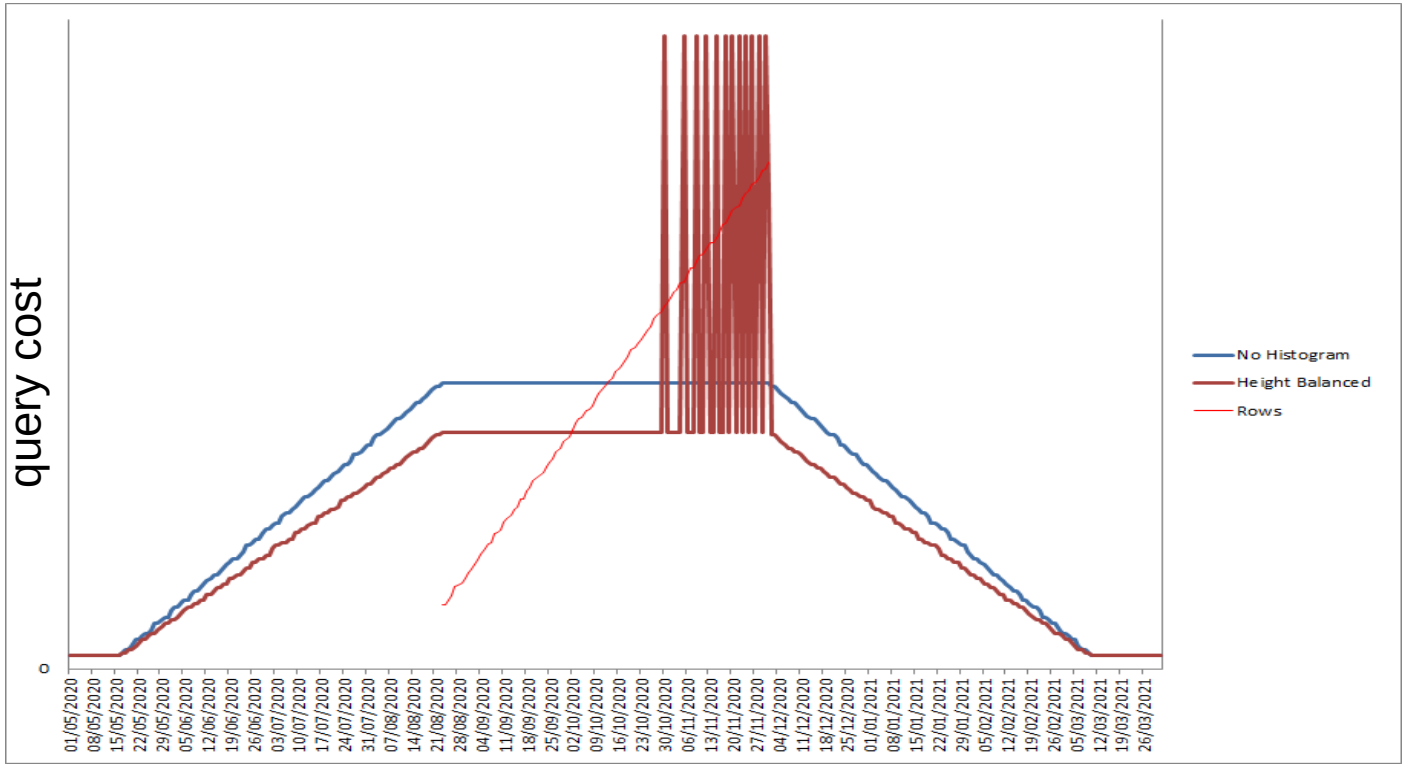
HISTOGRAM MYTHS

Top Frequency is really good too!



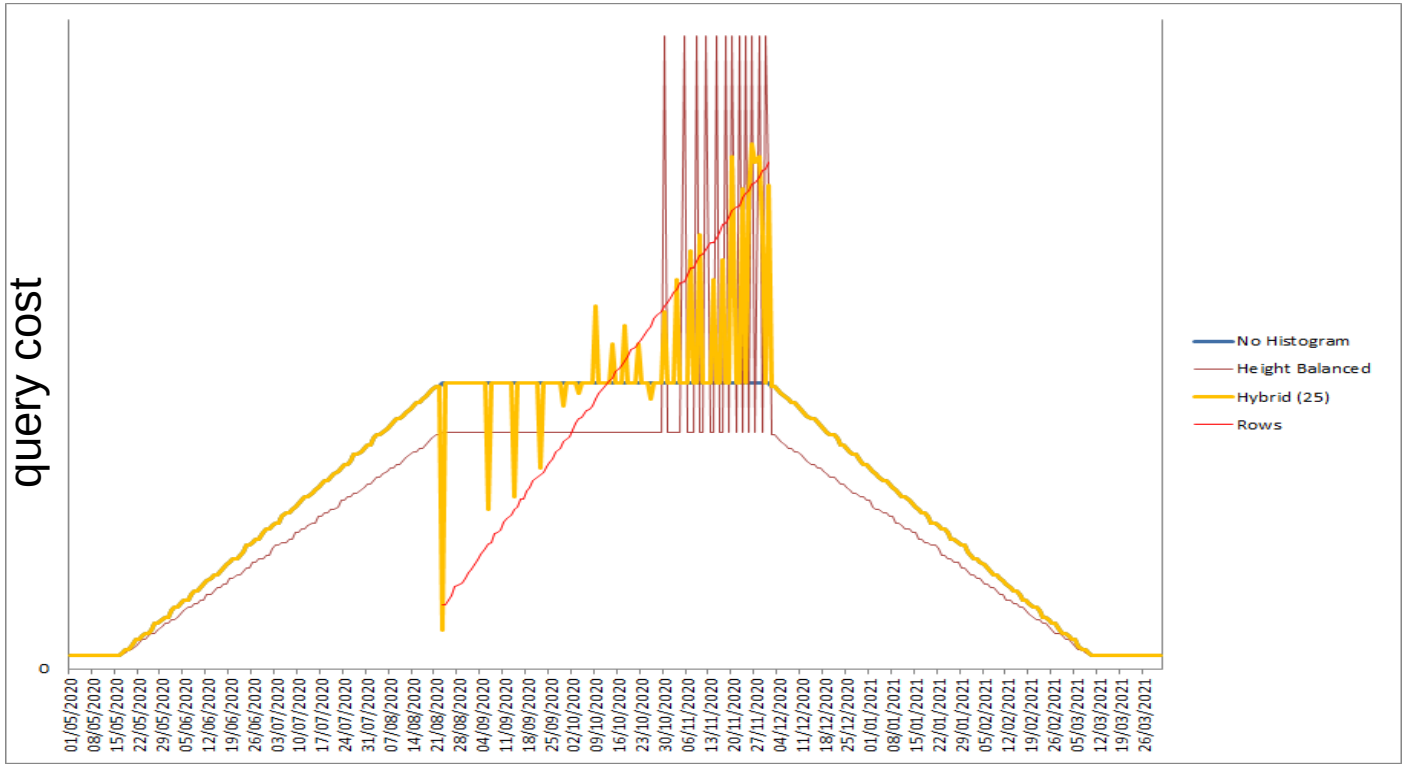
HISTOGRAM MYTHS

Height-Balanced - not as bad as you first think!



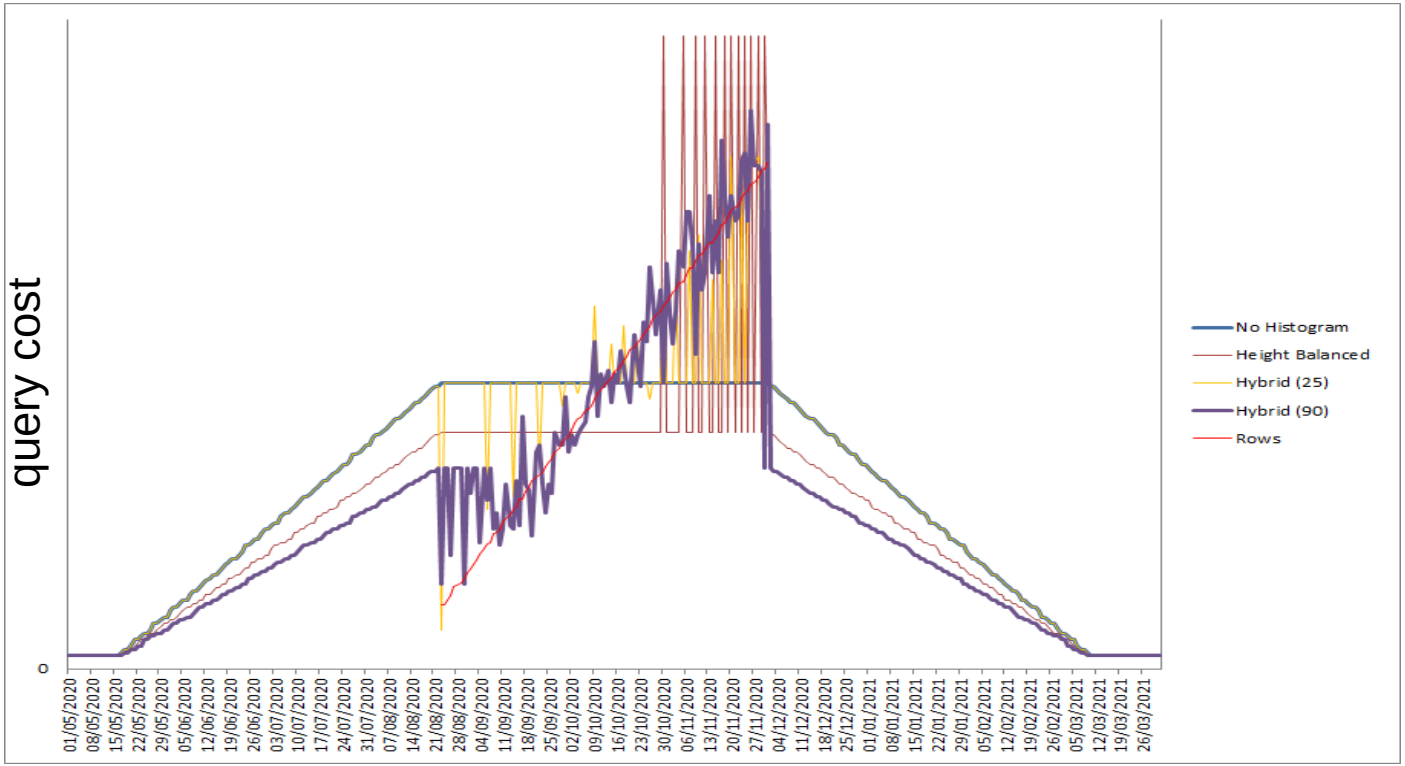
HISTOGRAM MYTHS

Aren't Hybrid histograms good!(25% NDV's)



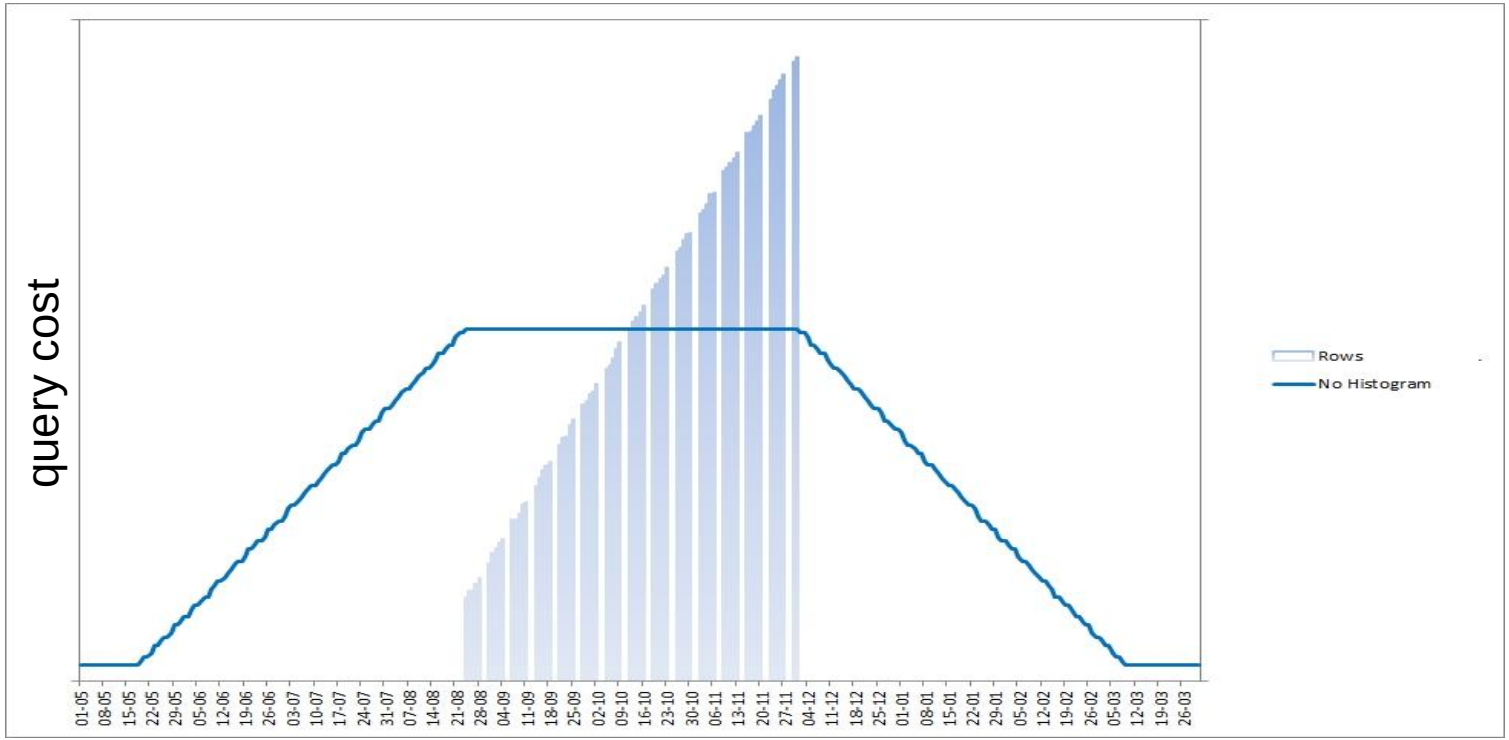
HISTOGRAM MYTHS

And even better with 90% of the NDV's captured



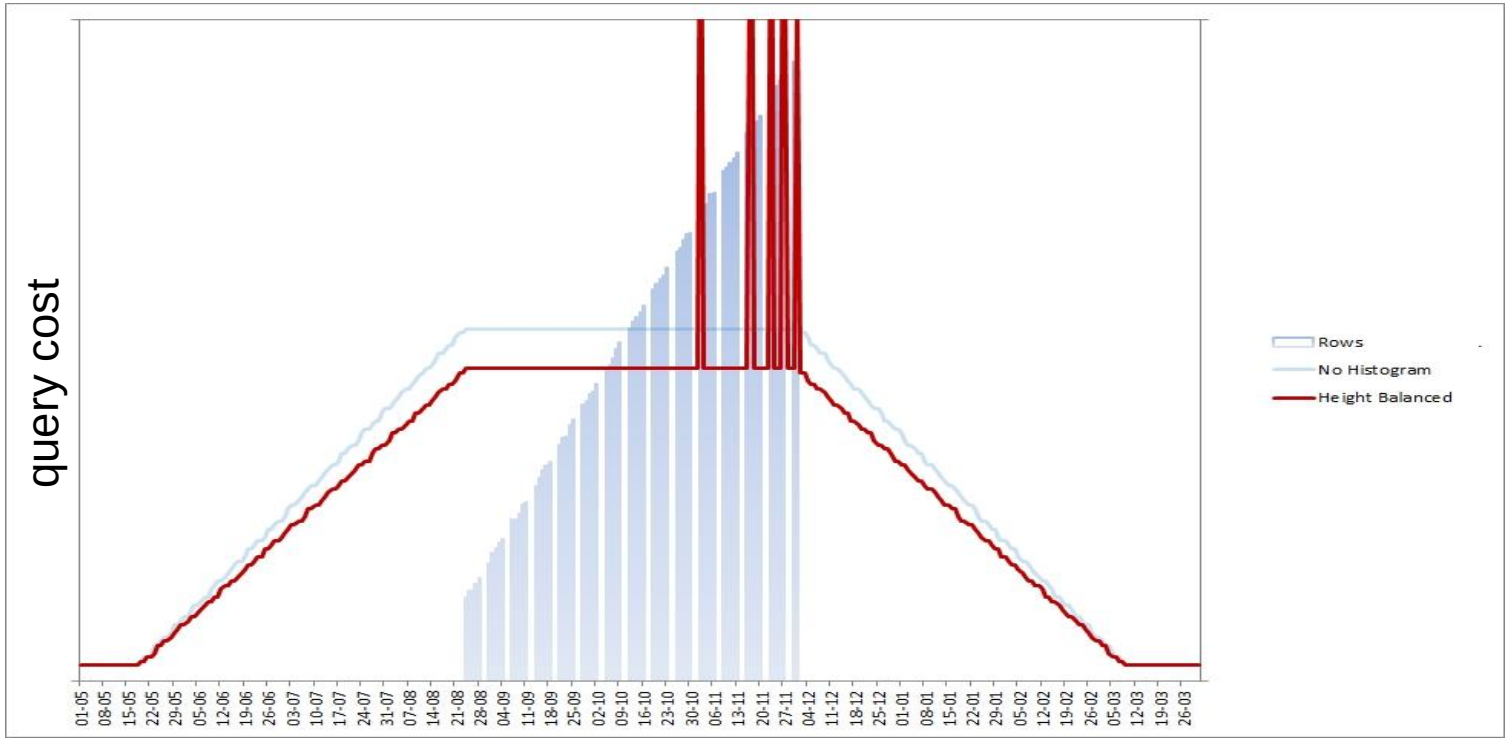
HISTOGRAM MYTHS

No Histogram, with gaps in the data



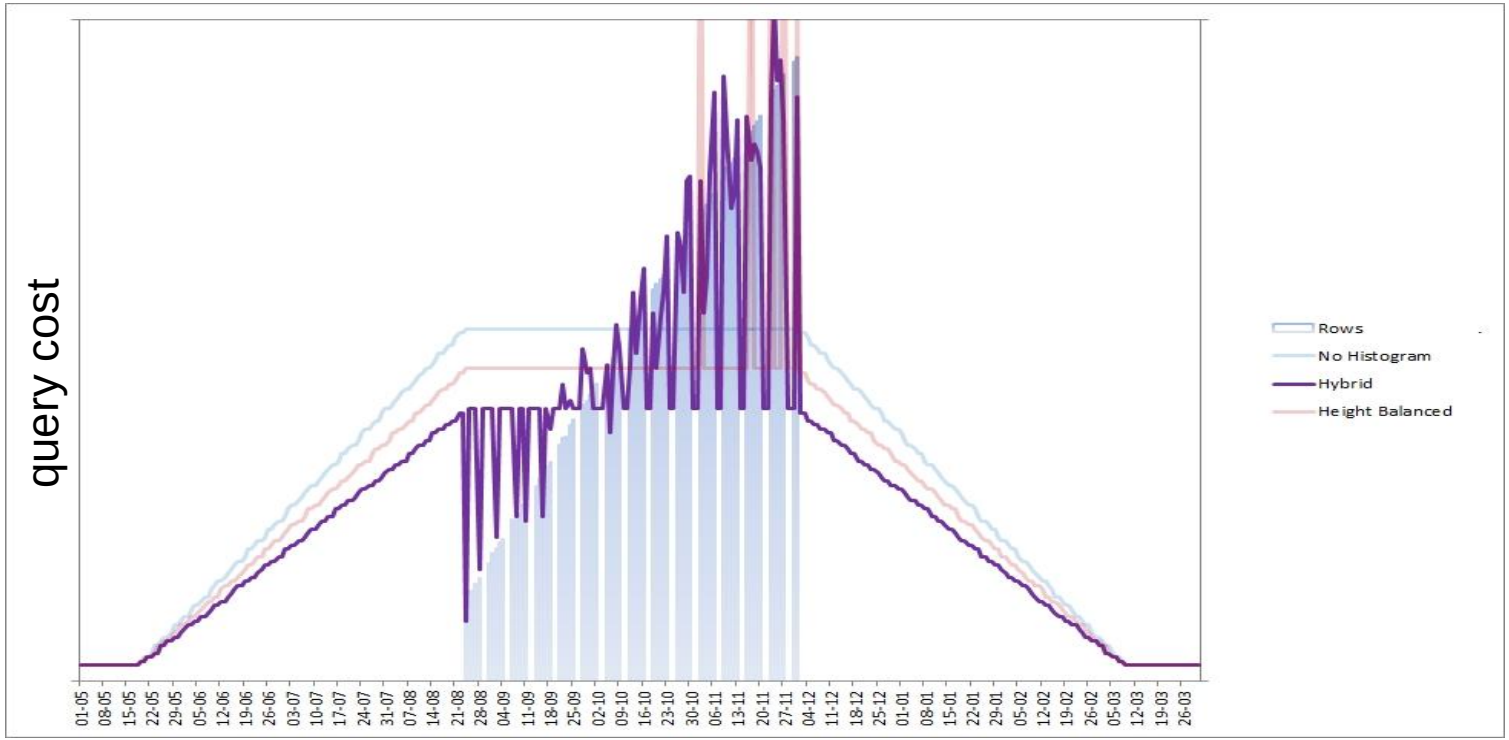
HISTOGRAM MYTHS

Height Balanced (90%) with gaps in the data



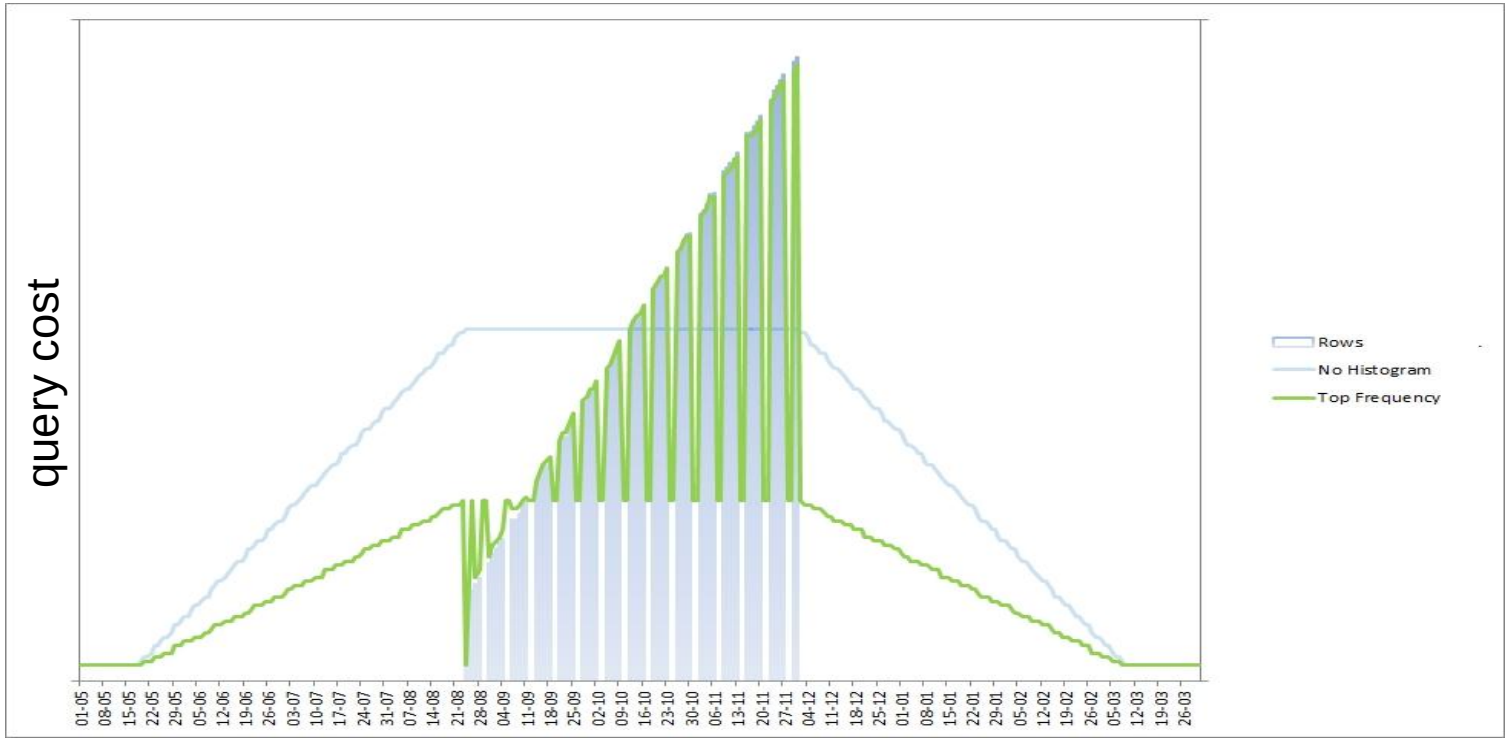
HISTOGRAM MYTHS

Hybrid (90%) with gaps in the data



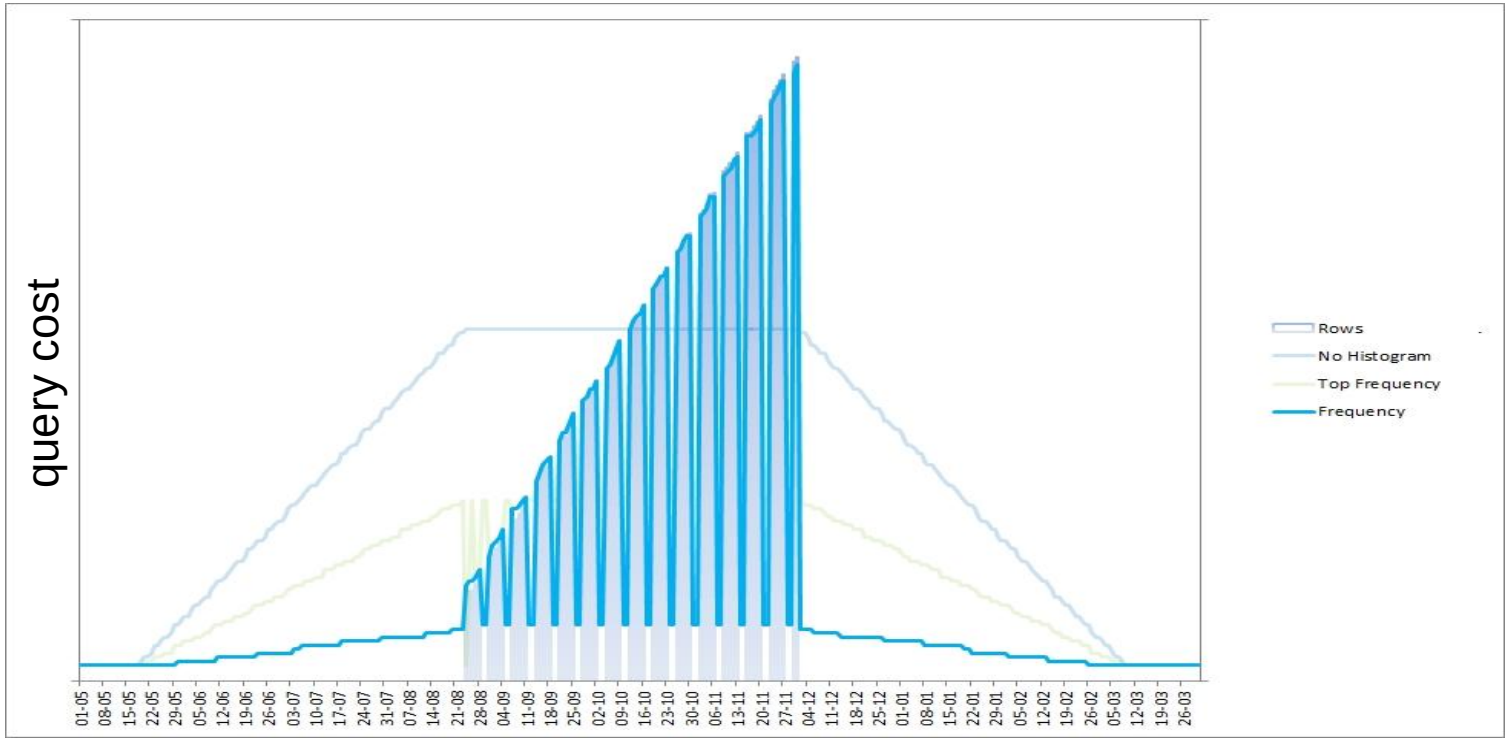
HISTOGRAM MYTHS

Top Frequency with gaps in the data



HISTOGRAM MYTHS

Frequency with gaps in the data



WHAT TO DO?

so what should we do to minimise the threats?

SIZE REPEAT

minimise the number of histograms?

FOR ALL COLUMNS SIZE REPEAT

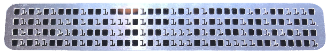
SIZE REPEAT

```
exec dbms_stats.set_global_prefs
('method_opt','for all columns size repeat');
```

This was a **great** idea in Oracle 11.2 and earlier.
It stopped more Histograms appearing, but maintained the ones currently in place.

Oracle have changed the algorithm in 12C

I do not like this change



SIZE REPEAT

Official Documentation for this feature

- REPEAT : Collects histograms only on the columns that already have histograms

Maria Colgan (now Nigel Bayliss) Blog from April 2013

https://blogs.oracle.com/optimizer/how-does-the-methodopt-parameter-work

REPEAT ensures a histogram will only be created for any column that already has one. If the table is a partitioned table, repeat ensures a histogram will be created for a column that already has one on the global level. However, this is not a recommended setting, as the number of buckets currently in each histogram will limit the maximum number of buckets used for the newly created histograms. Lets assume there are 5 buckets currently in a histogram. When the histogram is re-gathered with SIZE REPEAT, the newly created histogram will use at most 5 buckets and may not been of good quality.

(NOTE: it does not reference versions)

Oracle 12.1 was released in June 2013 and therefore this documents *future behaviours*

SIZE REPEAT

```
create table TEST_REPEAT (c1 number not null);
insert into  TEST_REPEAT (c1) select mod (rownum,20)+1 from dba_objects where rownum < 1001;
```

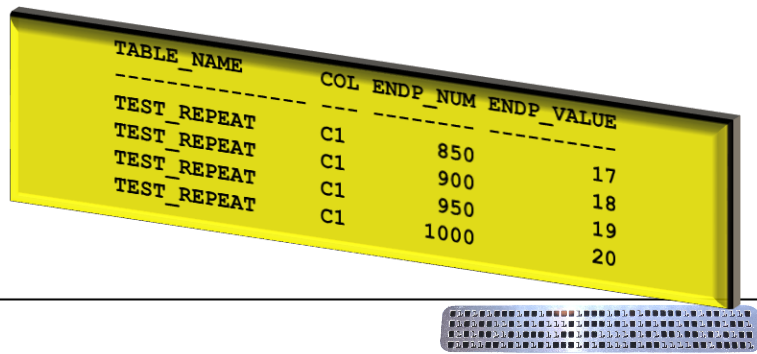
gather_table_stats

DBA_TAB_COLUMNS

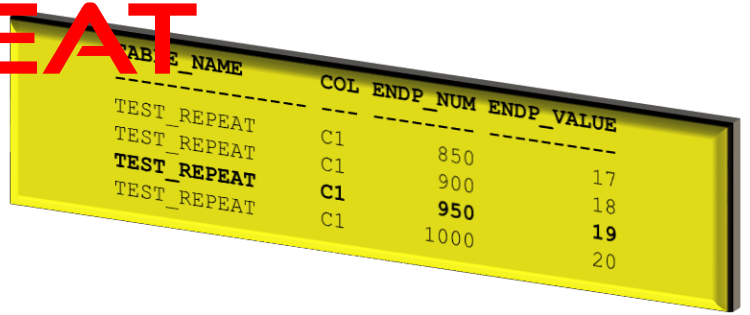
TABLE_NAME	COLUMN_NAM	NUM_DISTINCT	NUM_BUCKETS	HISTOGRAM
TEST_REPEAT	C1	20	20	FREQUENCY

DBA_TAB_HISTOGRAMS

TABLE_NAME	COLUMN_NAM	ENDPOINT_NUMBER	ENDPOINT_VALUE
.	.	.	.
TEST_REPEAT	C1	850	17
TEST_REPEAT	C1	900	18
TEST_REPEAT	C1	950	19
TEST_REPEAT	C1	1000	20



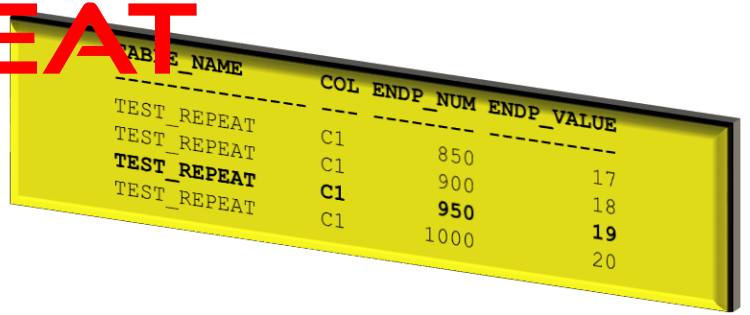
SIZE REPEAT



11.2

```
delete from TEST_REPEAT where c1 = 19;

exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',
     method_opt=>'FOR ALL COLUMNS SIZE REPEAT');

TABLE_NAME      COLUMN_NAM NUM_DISTINCT NUM_BUCKETS HISTOGRAM
--------------- ---------- ------------ ----------- ---------------
TEST_REPEAT     C1                   19          19 FREQUENCY

insert into TEST_REPEAT (c1) values (19);

TABLE_NAME      COLUMN_NAM NUM_DISTINCT NUM_BUCKETS HISTOGRAM
--------------- ---------- ------------ ----------- ---------------
TEST_REPEAT     C1                   20          20 FREQUENCY

insert into TEST_REPEAT (c1) values (21);

TABLE_NAME      COLUMN_NAM NUM_DISTINCT NUM_BUCKETS HISTOGRAM
--------------- ---------- ------------ ----------- ---------------
TEST_REPEAT     C1                   21          21 FREQUENCY
```

SIZE REPEAT



12.2

```
delete from TEST_REPEAT where c1 = 19;

exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',
     method_opt=>'FOR ALL COLUMNS SIZE REPEAT');

TABLE_NAME      COLUMN_NAM NUM_DISTINCT NUM_BUCKETS HISTOGRAM
--------------- ---------- ------------ ----------- ---------------
TEST_REPEAT     C1                   19          19 FREQUENCY

insert into TEST_REPEAT (c1) values (19);

TABLE_NAME      COLUMN_NAM NUM_DISTINCT NUM_BUCKETS HISTOGRAM
--------------- ---------- ------------ ----------- ---------------
TEST_REPEAT     C1                   20          19 TOP-FREQUENCY

insert into TEST_REPEAT (c1) values (21);

TABLE_NAME      COLUMN_NAM NUM_DISTINCT NUM_BUCKETS HISTOGRAM
--------------- ---------- ------------ ----------- ---------------
TEST_REPEAT     C1                   21          19 HYBRID
```

SIZE REPEAT CODE

```
column table_name  format a15
column column_name format a10
column low_value   format a10
column high_value  format a10

drop   table TEST_REPEAT purge;

set echo on
create table TEST_REPEAT (c1 number not null);

insert into TEST_REPEAT (c1) select mod (rownum,20)+1 from dba_objects where rownum < 1001;
commit;

prompt Need a repeated query before size auto kicks in for a histogram
select count(*) from TEST_REPEAT where c1 > 4;
select count(*) from TEST_REPEAT where c1 > 4;

exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',method_opt=>'FOR ALL COLUMNS SIZE AUTO');
exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',method_opt=>'FOR ALL COLUMNS SIZE REPEAT');

select table_name,column_name,num_distinct,num_buckets,histogram from dba_tab_columns where table_name = 'TEST_REPEAT';
select TABLE_NAME,COLUMN_NAME,ENDPOINT_NUMBER,ENDPOINT_VALUE from user_tab_histograms where endpoint_value > 16 order by TABLE_NAME,ENDPOINT_NUMBER;

delete from TEST_REPEAT where c1 = 19;
commit;
exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',method_opt=>'FOR ALL COLUMNS SIZE REPEAT');
select table_name,column_name,num_distinct,num_buckets,histogram from dba_tab_columns where table_name = 'TEST_REPEAT';
select TABLE_NAME,COLUMN_NAME,ENDPOINT_NUMBER,ENDPOINT_VALUE from user_tab_histograms where endpoint_value > 16 order by TABLE_NAME,ENDPOINT_NUMBER;

insert into TEST_REPEAT (c1) values (19);
commit;
exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',method_opt=>'FOR ALL COLUMNS SIZE REPEAT');
select table_name,column_name,num_distinct,num_buckets,histogram from dba_tab_columns where table_name = 'TEST_REPEAT';
select TABLE_NAME,COLUMN_NAME,ENDPOINT_NUMBER,ENDPOINT_VALUE from user_tab_histograms where endpoint_value > 16 order by TABLE_NAME,ENDPOINT_NUMBER;

insert into TEST_REPEAT (c1) values (21);
commit;
exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',method_opt=>'FOR ALL COLUMNS SIZE REPEAT');
select table_name,column_name,num_distinct,num_buckets,histogram from dba_tab_columns where table_name = 'TEST_REPEAT';
select TABLE_NAME,COLUMN_NAME,ENDPOINT_NUMBER,ENDPOINT_VALUE from user_tab_histograms where endpoint_value > 16 order by TABLE_NAME,ENDPOINT_NUMBER;

exec dbms_stats.gather_table_stats (null,'TEST_REPEAT',method_opt=>'FOR ALL COLUMNS SIZE AUTO');
select table_name,column_name,num_distinct,num_buckets,histogram from dba_tab_columns where table_name = 'TEST_REPEAT';
select TABLE_NAME,COLUMN_NAME,ENDPOINT_NUMBER,ENDPOINT_VALUE from user_tab_histograms where endpoint_value > 16 order by TABLE_NAME,ENDPOINT_NUMBER;

set echo off
```

Don't believe me? Try it yourself.

DO NOT USE: SIZE REPEAT

```
method_opt: for all columns size repeat
```

should NOT be used in ORACLE 12C

so what should we do?

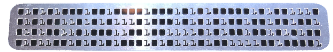
CONCLUSION : BA

for
Data Warehouse Systems
Business Intelligence Systems
Data Analytics Systems
where unique large and complex queries are common

Use Histograms

(well… except Height-Balanced)

It is *w*orth specific intervention on the
BIG FACT TABLES
to optimize resource usage

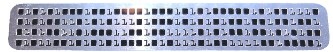


CONCLUSION : OLTP

for **OLTP** systems
where performance *consistency* is critical
consider the following:

switch off histograms globally (on new systems!)

use **SET_TABLE_PREFS**
to control histograms for individual columns



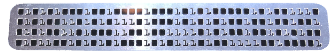
CONCLUSION : OLTP

How do I control histograms?

Switch off Globally (on new systems!)

```
exec dbms_stats.set_global_prefs(
    'method_opt',
    'for all columns size 1');
```

Do NOT do this on your existing Production system!



CONCLUSION : OLTP

How do I control histograms?

Determine where you have CARDINALITY problems and threats

Look at ESTIMATE vs ACTUAL issues in your explain plans

```
DBMS_XPLAN - Execution Plan

select ID from test_basic where status = 'ERROR';
-------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name       | Starts | E-Rows |E-Bytes|Cost (%CPU)|A-Rows |Buffers|
-------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |            |      1 |        |       | 171 (100) |     5 |   570 |
|*  1 |  TABLE ACCESS FULL                 | TEST_BASIC |      1 |  66672 |   911K| 171   (1) |     5 |   570 |
-------------------------------------------------------------------------------------------------------------------
```

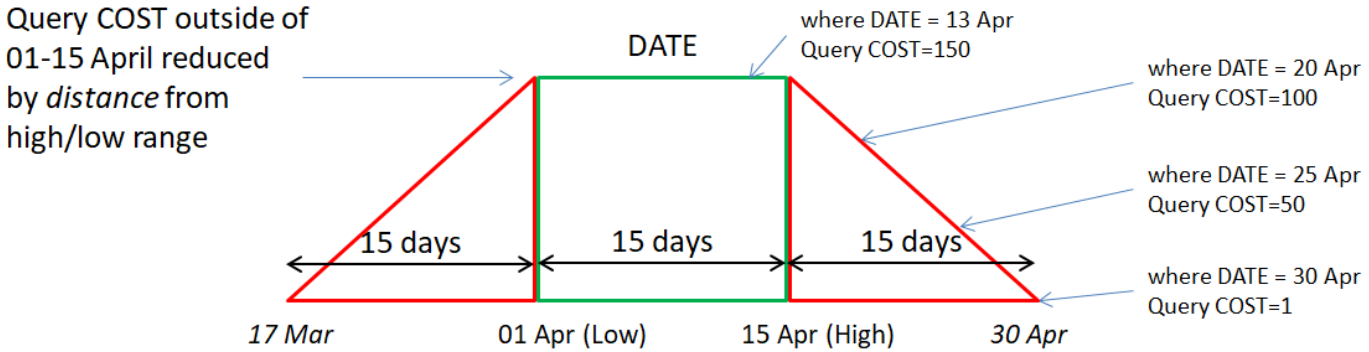
Look to see if you are getting PLAN STABILITY issues (plan flips) where the columns have HYBRID and HEIGHT-BALANCED histograms and consider removing them

CONCLUSION : OLTP

How do I control histograms?

Determine where you have CARDINALITY problems and threats

Determine where Frequency and Top Frequency Histograms have Low/High-Value Threats
(where you are querying above or below the Low or High values)



CONCLUSION : OLTP

How do I control histograms?

Set Specific Cases for each table using SET_TABLE_PREFS

```
DBMS_STATS.SET_TABLE_PREFS(ownname=>'NEIL', tabname=>'TEST_REPEAT', pname  =>'method_opt',
     pvalue =>'for all columns size 1
               for columns size auto col1,col2,col3');
```

Consider OVERRIDING command-line Params so the SET_TABLE_PREFS is *always* used (12.2)

```
DBMS_STATS.SET_TABLE_PREFS('NEIL','TEST_REPEAT', 'PREFERENCE_OVERRIDES_PARAMETER','TRUE')
```

USER_TAB_STAT_PREFS

```
TABLE_NAME      PREFERENCE_NAME                PREFERENCE_VALUE
--------------- ------------------------------ ------------------------------------------------------------
TEST_REPEAT     METHOD_OPT                     FOR ALL COLUMNS SIZE 1 FOR COLUMNS SIZE AUTO col1,col2,col3
TEST_REPEAT     PREFERENCE_OVERRIDES_PARAMETER TRUE
```


THANKS

Thanks to the knowledge, research, blogs and white papers of:

- Maria Colgan
- The rest of the core Ask ToM team (Chris and Connor)
- Nigel Bayliss
- Jonathan Lewis
- Amit Poddar (One Pass Distinct Sampling Paper and Presentation)
- Christian Antognini
- Mohamed Houri
- Wolfgang Breitling
- Probably Martin Widlake too
- anyone else who has blogged or presented about Histograms, Approximate NDV or any other related subject I've accidentally ingested over the years

and

- The Oracle Manuals

ANY QUESTIONS

BLOG: http://chandlerdba.wordpress.com/

Email: neil@chandler.uk.com

Twit: @chandlerdba