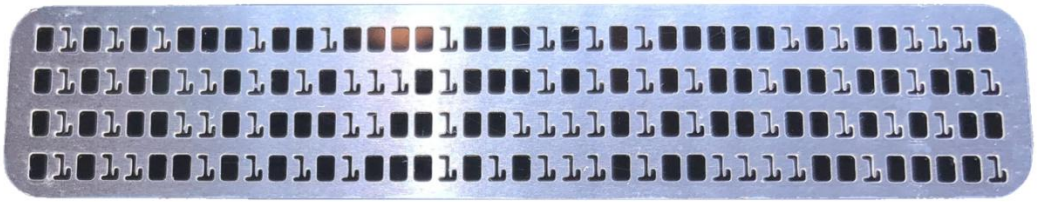
STATISTICS

DOING IT RIGHT, THE EASY WAY

Neil Chandler ORACLE ACE Director

Chandler Systems



Independent Database Consultant

BLOG: http://chandlerdba.com

Twitter: **@chandlerDBA**

E: **neil@chandler.uk.com**

VIDEO

RULE BASED OPTIMISER (RBO)

1. Deprecated in Oracle 10G, but still used by Oracle themselves
2. Didn't know anything about the shape of your data
3. Followed rules based upon how you coded your SQL, plus some **heuristics** (rules) for the access path
4. Probably the source of the "Full Table Scan is Bad" myth

(published) Rank of RBO Access Path Heuristics ("best" to "worst")

RBO Path 1: Single Row by Rowid
RBO Path 2: Single Row by Cluster Join
RBO Path 3: Single Row by Hash Cluster Key with Unique or Primary Key
RBO Path 4: Single Row by Unique or Primary Key
RBO Path 5: Clustered Join
RBO Path 6: Hash Cluster Key
RBO Path 7: Indexed Cluster Key
RBO Path 8: Composite Index
RBO Path 9: Single-Column Indexes
RBO Path 10: Bounded Range Search on Indexed Columns
RBO Path 11: Unbounded Range Search on Indexed Columns
RBO Path 12: Sort Merge Join
RBO Path 13: MAX or MIN of Indexed Column
RBO Path 14: ORDER BY on Indexed Column
RBO Path 15: Full Table Scan

RULE BASED OPTIMISER (RBO)

1. Deprecated in Oracle 10G, but still used by Oracle themselves
2. Didn't know anything about the shape of your data
3. Followed rules based upon how you coded your SQL, plus some heuristics (rules) for the access path
4. Probably the source of the "Full Table Scan is Bad" myth
5. Still used (a little bit) by Oracle when accessing the Data Dictionary

from v$sql in a 19C database:

```
select /*+ rule */ bucket_cnt,
row_cnt, cache_cnt, null_cnt,
 timestamp#, sample_size, minimum,
maximum, distcnt, lowval,
 hival, density, col#, spare1,
spare2, avgcln, minimum_enc,
maximum_enc from hist_head$ where
obj#=:1 and intcol#=:2
```

(published) Rank of RBO Access Path Heuristics ("best" to "worst")

RBO Path 1: Single Row by Rowid
RBO Path 2: Single Row by Cluster Join
RBO Path 3: Single Row by Hash Cluster Key with Unique or Primary Key
RBO Path 4: Single Row by Unique or Primary Key
RBO Path 5: Clustered Join
RBO Path 6: Hash Cluster Key
RBO Path 7: Indexed Cluster Key
RBO Path 8: Composite Index
RBO Path 9: Single-Column Indexes
RBO Path 10: Bounded Range Search on Indexed Columns
RBO Path 11: Unbounded Range Search on Indexed Columns
RBO Path 12: Sort Merge Join
RBO Path 13: MAX or MIN of Indexed Column
RBO Path 14: ORDER BY on Indexed Column
RBO Path 15: Full Table Scan

RULE BASED OPTIMISER (RBO)

1. Deprecated in Oracle 10G, but still used by Oracle themselves
2. Didn't know anything about the shape of your data
3. Followed rules based upon how you coded your SQL, plus some heuristics for the access path
4. Probably the source of the "Full Table Scan is Bad" myth
5. Still used (a little bit) by Oracle when accessing the Data Dictionary

from v$sql in a 10g database:

```
select /*+ rule */ bucket_cnt,
row_cnt, cache_cnt, null_cnt,
 timestamp#, sample_size, minimum,
maximum, distcnt, lowval,
 hival, density, col#, spare1,
spare2, avgcln, minimum_enc,
maximum_enc from hist_head$ where
obj#=:1 and intcol#=:2
```

Performance Problem Querying the Data Dictionary?

try the /*+ RULE */ hint

(published) Rank of RBO Access Path Heuristics ("best" to "worst")

RBO Path 1: Single Row by Rowid
RBO Path 2: Single Row by Cluster Join
RBO Path 3: Single Row by Hash Cluster Key with Unique or Primary Key
RBO Path 4: Single Row by Unique or Primary Key
RBO Path 5: Clustered Join
RBO Path 6: Hash Cluster Key
RBO Path 7: Indexed Cluster Key
RBO Path 8: Composite Index
RBO Path 9: Single-Column Indexes
RBO Path 10: Bounded Range Search on Indexed Columns
RBO Path 11: Unbounded Range Search on Indexed Columns
RBO Path 12: Sort Merge Join
RBO Path 13: MAX or MIN of Indexed Column
RBO Path 14: ORDER BY on Indexed Column
RBO Path 15: Full Table Scan

RULE BASED OPTIMISER (RBO)

1. Deprecated in Oracle 10G, but still use
2. Didn't know anything about the shape of your data
3. Followed rules based upon how you coded your SQL
   plus some heuristics for the access path
4. Probably the source of the "Full Table Scans Bad" myth
5. Still used (a little bit) by Oracle when
   accessing the Data Dictionary

BUT I've seen the /*+ RULE */ hint, in application SQL, in 2020, in Oracle 19.7 databases. HINTS are hard to remove….. but the Optimizer is helping:

Hint Report
Total hints for statement:1 (U - Unused (1))
0 - STATEMENT
U - RULE

from v$sql in a 10g database:

```
select /*+ rule */ bucket_cnt,
row_cn  ount, null_cnt,
 timest  imp#, sample_size, minimum,
maximum  distcnt, lowval,
 hival,  ensity, col#, spare1,
spare2,  lgcln, minimum_ep,
maximum_  c from hist_head$ where
obj#=:1  and intcol#=:2
```

Performance Problem Querying Data Dictionary?

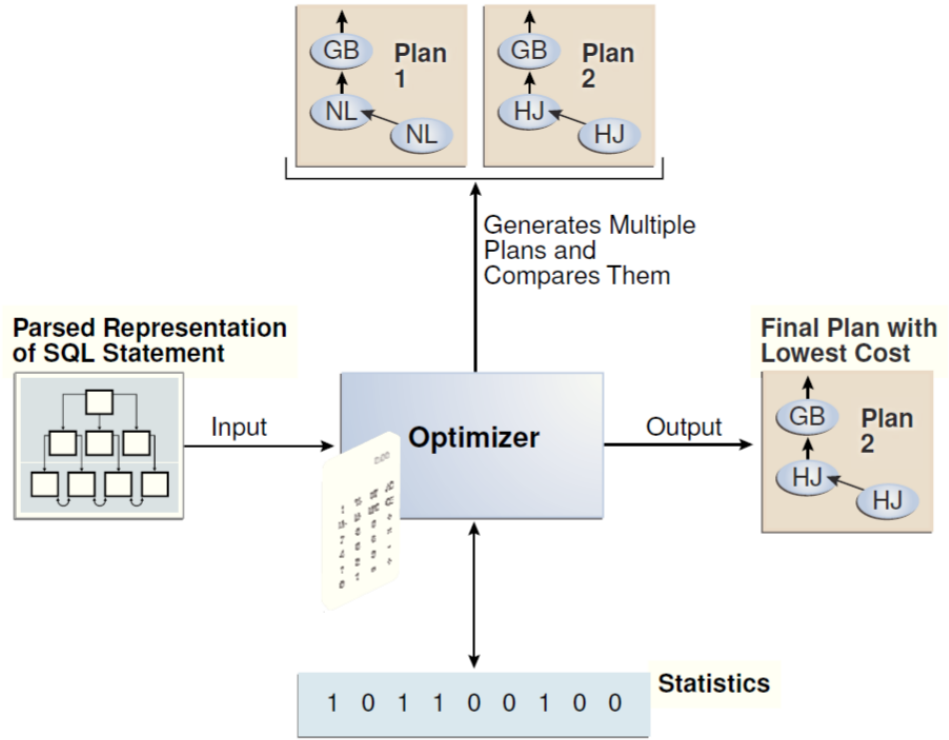
try the /*+ RULE */ hint

RBO Path 1: Single Row by Rowid
RBO Path 2: Single Row by Cluster Join
RBO Path 3: Single Row by Hash Cluster Key with Unique or Primary Key
RBO Path 4: Single Row by Unique or Primary Key
RBO Path 5: Clustered Join
RBO Path 6: Hash Cluster Key
RBO Path 7: Indexed Cluster Key
RBO Path 8: Composite Index
RBO Path 9: Single-Column Indexes
RBO Path 10: Bounded Range Search on Indexed Columns
RBO Path 11: Unbounded Range Search on Indexed Columns
RBO Path 12: Sort Merge Join
RBO Path 13: MAX or MIN of Indexed Column
RBO Path 14: ORDER BY on Indexed Column
RBO Path 15: Full Table Scan

COST-BASED OPTIMISER (CBO)

Diagram borrowed from Oracle 19c Tuning Guide

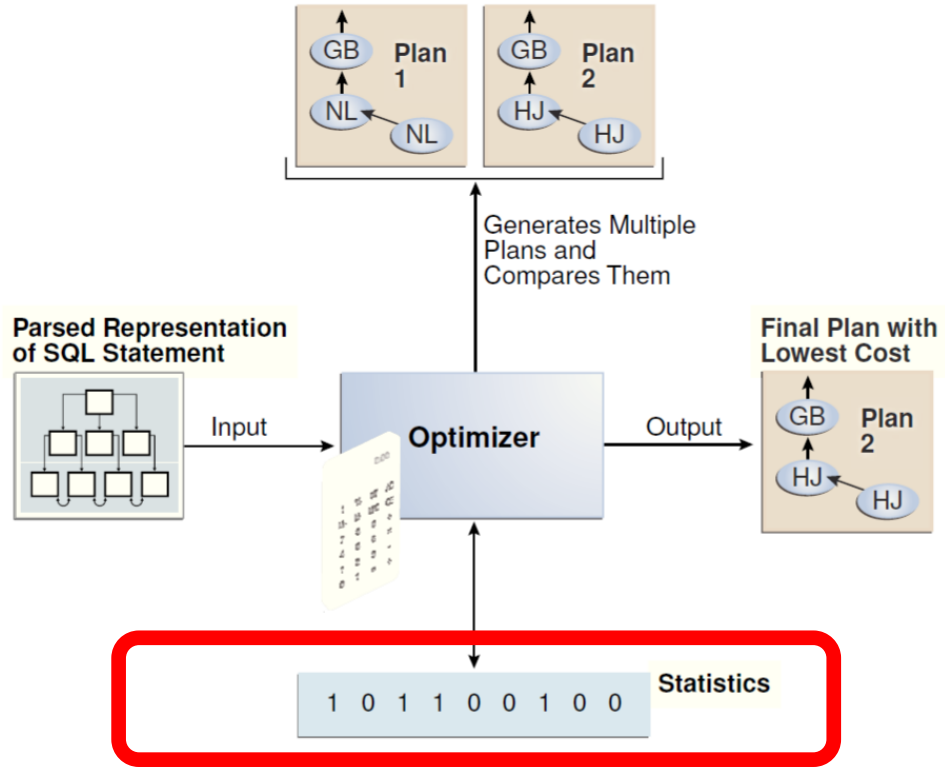
Contributing Authors: Nigel Bayliss, Maria Colgan, Tom Kyte



COST-BASED OPTIMISER (CBO)

Diagram borrowed from Oracle 19c Tuning Guide

Contributing Authors: Nigel Bayliss, Maria Colgan, Tom Kyte



WHAT ARE THESE STATISTICS ANYWAY?

DBA_TABLES
DBA_TAB_STATISTICS
DBA_TAB_PARTITIONS
DBA_TAB_SUB_PARTITIONS
DBA_TAB_COLUMNS
DBA_TAB_COL_STATISTICS
DBA_PART_COL_STATISTICS
DBA_SUBPART_COL_STATISTICS
DBA_INDEXES
DBA_IND_STATISTICS
DBA_IND_PARTITIONS
DBA_IND_SUBPARTIONS
DBA_TAB_HISTOGRAMS
DBA_PART_HISTOGRAMS
DBA_SUBPART_HISTOGRAMS

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_STATISTICS

OWNER	Owner of the object
TABLE_NAME	Name of the table
PARTITION_NAME	Name of the partition
PARTITION_POSITION	Position of the partition within table
SUBPARTITION_NAME	Name of the subpartition
SUBPARTITION_POSITION	Position of the subpartition within partition
OBJECT_TYPE	Type of the object (TABLE, PARTITION, SUBPARTITION)
NUM_ROWS	The number of rows in the object
BLOCKS	The number of used blocks in the object
EMPTY_BLOCKS	The number of empty blocks in the object
AVG_SPACE	The average available free space in the object
CHAIN_CNT	The number of chained rows in the object
AVG_ROW_LEN	The average row length, including row overhead
AVG_SPACE_FREELIST_BLOCKS	The average freespace of all blocks on a freelist
NUM_FREELIST_BLOCKS	The number of blocks on the freelist
AVG_CACHED_BLOCKS	Average number of blocks in buffer cache
AVG_CACHE_HIT_RATIO	Average cache hit ratio for the object
IM_IMCU_COUNT	Number of IMCUs in the object
IM_BLOCK_COUNT	Number of inmemory blocks in the object
IM_STAT_UPDATE_TIME	The timestamp most recent update to the inmemory stat
SCAN_RATE	Scan rate for the object
SAMPLE_SIZE	The sample size used in analyzing this table
LAST_ANALYZED	The date of the most recent time this table was analyzed
GLOBAL_STATS	Are the statistics calculated without merging underlying partitions?
USER_STATS	Were the statistics entered directly by the user?
STATTYPE_LOCKED	type of statistics lock
STALE_STATS	Whether statistics for the object is stale or not
NOTES	Notes regarding special properties of the stats
SCOPE	whether statistics for the object is shared or session

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_STATISTICS

NUM_ROWS	The number of rows in the object **= relative scale**
BLOCKS	The number of used blocks in the object **= I/O**
LAST_ANALYZED	When were the stats last gathered
SAMPLE_SIZE	The sample size used in stats gather

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_STATISTICS

NUM_ROWS	The number of rows in the object **= relative scale**
BLOCKS	The number of used blocks in the object **= I/O**
LAST_ANALYZED	When were the stats last gathered
SAMPLE_SIZE	The sample size used in stats gather
USER_STATS	Were the stats entered directly by the user?
STATTYPE_LOCKED	Locked stats aren't gathered automatically
STALE_STATS	Are the stats old and need re-gathering?
NOTES	New for 12C+
	Notes regarding special properties of the stats

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_**COL**_STATISTICS

NUM_DISTINCT The number of distinct values in the column

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column

LOW_VALUE The lowest value in the column

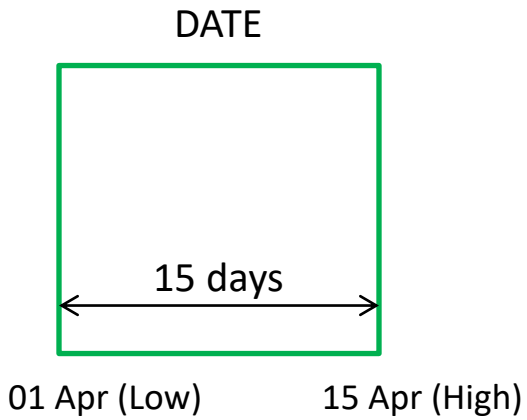
HIGH_VALUE The highest value in the column

Query costs lowered for query outside of these values through "statistical decay"

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column
LOW_VALUE The lowest value in the column
HIGH_VALUE The highest value in the column
Query costs lowered for query outside of these values through "statistical decay"



WHAT ARE THESE STATISTICS ANYWAY?

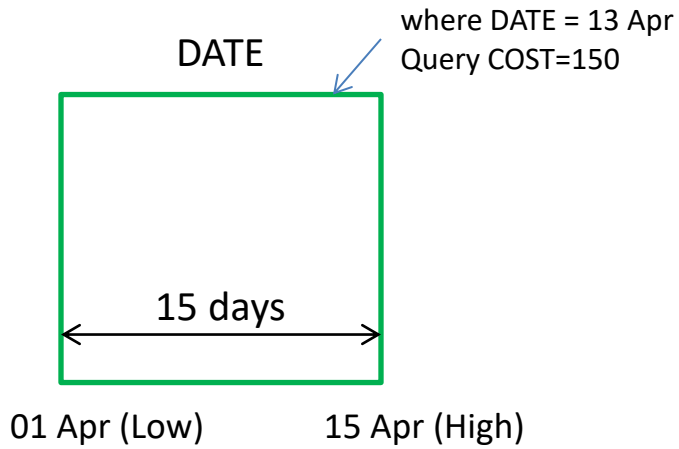
DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column

LOW_VALUE The lowest value in the column

HIGH_VALUE The highest value in the column

Query costs lowered for query outside of these values through "statistical decay"

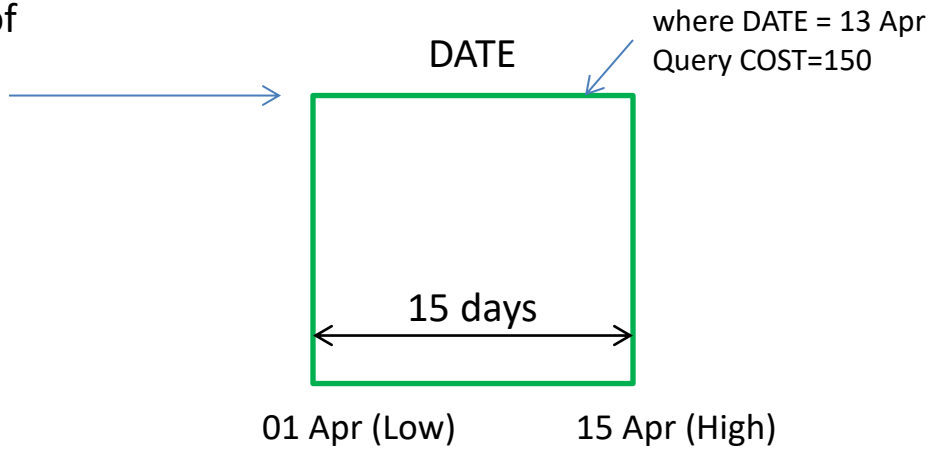


WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column
LOW_VALUE The lowest value in the column
HIGH_VALUE The highest value in the column
Query costs lowered for query outside of these values through "statistical decay"

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

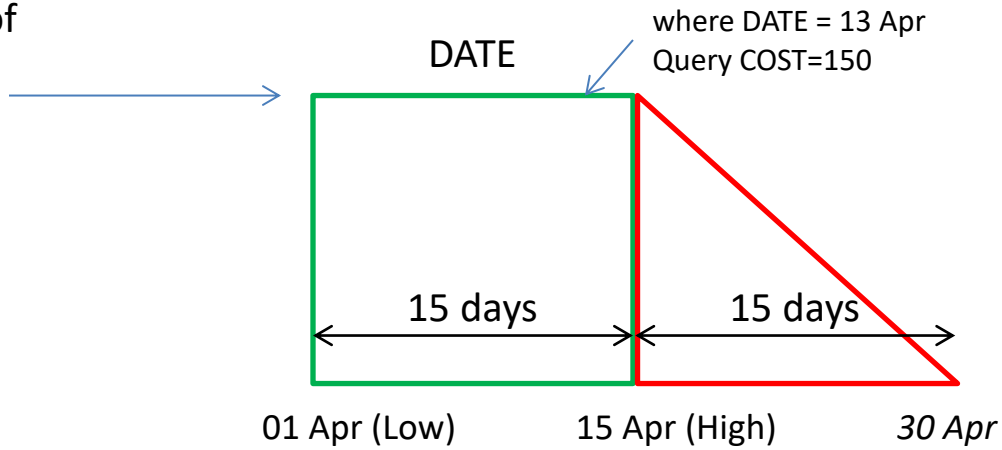


WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column
LOW_VALUE The lowest value in the column
HIGH_VALUE The highest value in the column
Query costs lowered for query outside of these values through "statistical decay"

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

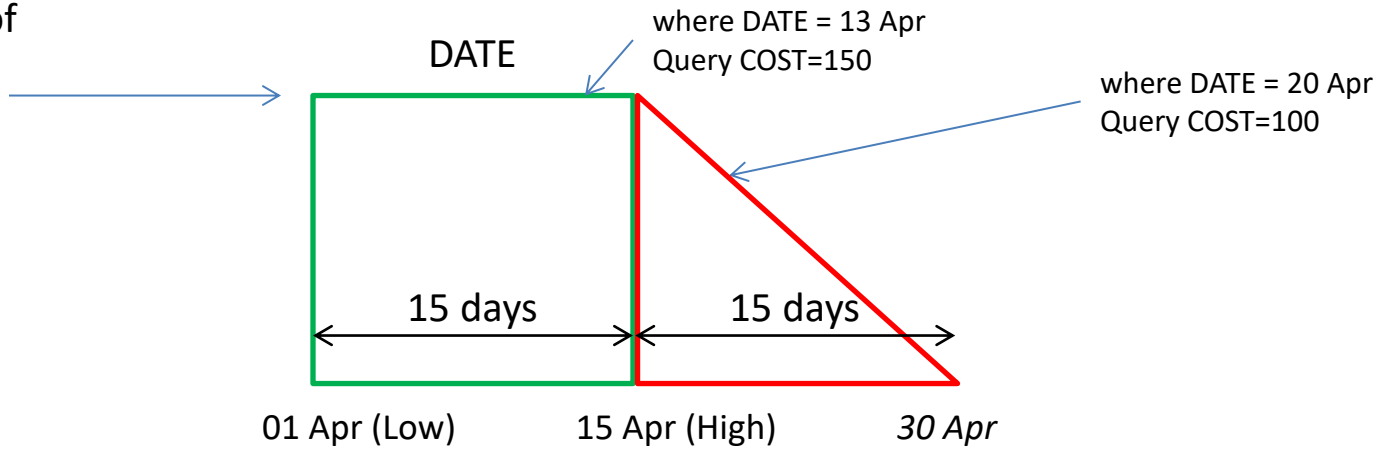


WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column
LOW_VALUE The lowest value in the column
HIGH_VALUE The highest value in the column
Query costs lowered for query outside of these values through "statistical decay"

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

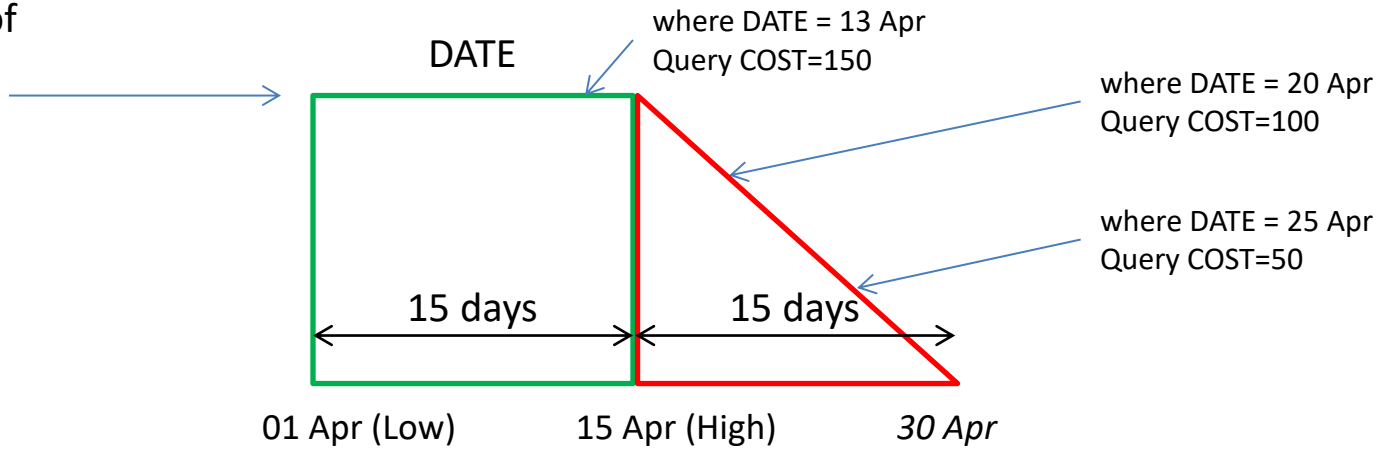


WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT The number of distinct values in the column
LOW_VALUE The lowest value in the column
HIGH_VALUE The highest value in the column
Query costs lowered for query outside of these values through "statistical decay"

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



WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

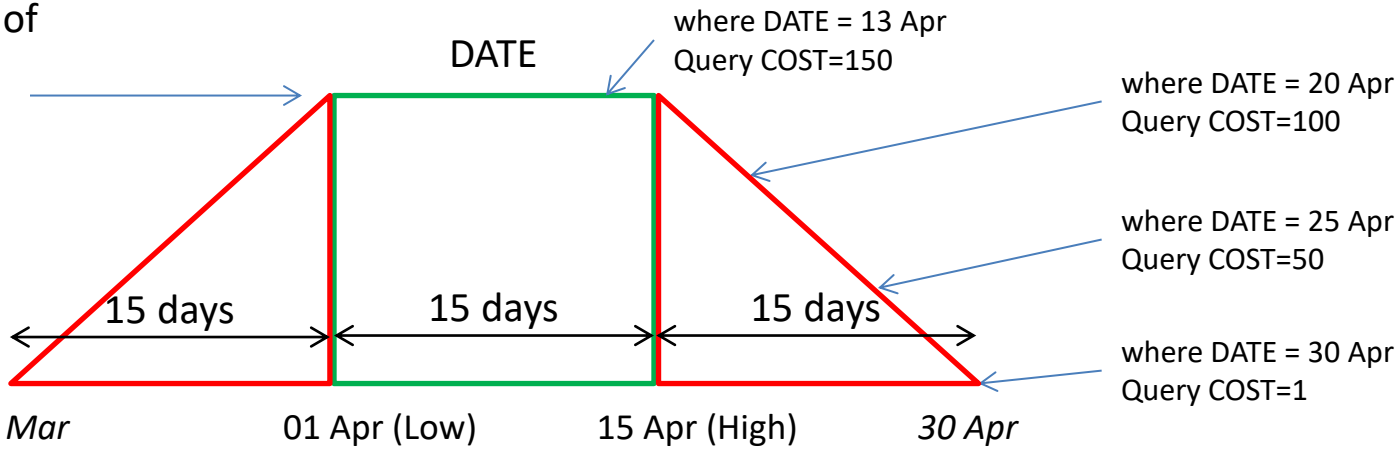
NUM_DISTINCT The number of distinct values in the column

LOW_VALUE The lowest value in the column

HIGH_VALUE The highest value in the column

Query costs lowered for query outside of these values through "statistical decay"

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



WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT	The number of distinct values in the column
LOW_VALUE	The lowest value in the column
HIGH_VALUE	The highest value in the column
	Query costs lowered for query outside of these values through "statistical decay"

NOTE: The DECAY reduces ***CARDINALITY*** - the amount of expected rows for a given predicate.

This influences the **COST** (how many I/O we will need to do) and which **ACCESS PATH** and **JOIN TYPES** will be used

TE = 20 Apr
ST=100

TE = 25 Apr
ST=50

TE = 30 Apr
OST=1

17 Mar 01 Apr (Low) 15 Apr (High) *30 Apr*

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT	The number of distinct values in the column
LOW_VALUE	The lowest value in the column
HIGH_VALUE	The highest value in the column
	Query costs lowered for query outside of these values
DENSITY	1/NUM_DISTINCT **= % of table retrieved for any specific value** (ignored if you have histogram)
NUM_NULLS	The number of nulls in the column

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_COL_STATISTICS

NUM_DISTINCT	The number of distinct values in the column
LOW_VALUE	The lowest value in the column
HIGH_VALUE	The highest value in the column
	Query costs lowered for query outside of these values
DENSITY	1/NUM_DISTINCT **= % of table retrieved for any specific value** (ignored if you have histogram)
NUM_NULLS	The number of nulls in the column
SAMPLE_SIZE	The sample size used in analyzing this column
HISTOGRAM	What type of histogram on the column
NUM_BUCKETS	The number of buckets in histogram for the column

WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

NUM_ROWS The number of rows in the index **= SCALE**

LEAF_BLOCKS The number of leaf blocks in the index **= I/O**

DISTINCT_KEYS Derive % of index to be used (NUM_ROWS/DISTINCT_KEYS)

CLUSTERING_FACTOR How aligned is the index to the **table?**

…

WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

NUM_ROWS	The number of rows in the index **= SCALE**
LEAF_BLOCKS	The number of leaf blocks in the index **= I/O**
DISTINCT_KEYS	Derive % of index to be used (NUM_ROWS/DISTINCT_KEYS)
CLUSTERING_FACTOR	How aligned is the index to the **table?**
	Can we re-read the **same** table block for the **next** index value?
	…

WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

NUM_ROWS	The number of rows in the index = SCALE
LEAF_BLOCKS	The number of leaf blocks in the index = I/O
DISTINCT_KEYS	Derive % of index to be used (NUM_ROWS/DISTINCT_KEYS)
CLUSTERING_FACTOR	How aligned is the index to the **table?**

Can we re-read the **same** table block for the **next** index value?

If the block is within the last **X** number of blocks, don't increase the Clustering Factor

X is determined by **TABLE_CACHED_BLOCKS**(default 1)

WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:

INDEX BLOCK 1:
row1
row2
row3
row4
row5

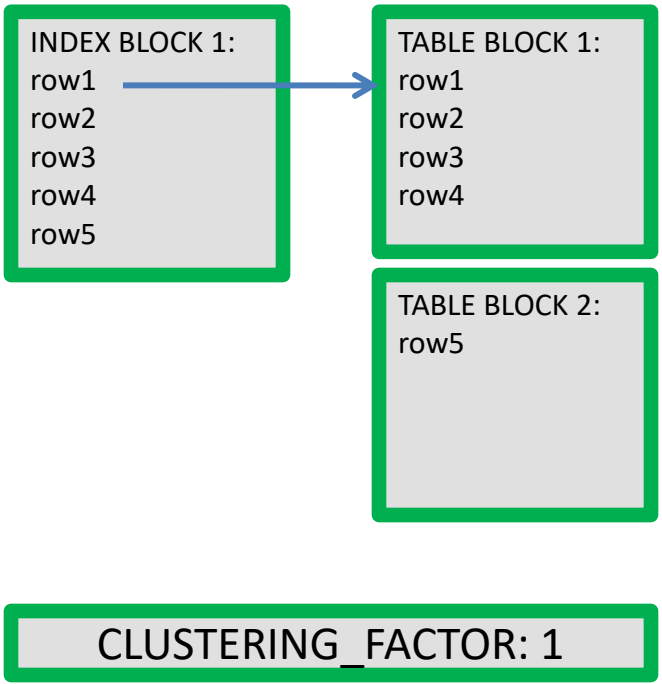
TABLE BLOCK 1:
row1
row2
row3
row4

TABLE BLOCK 2:
row5

WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

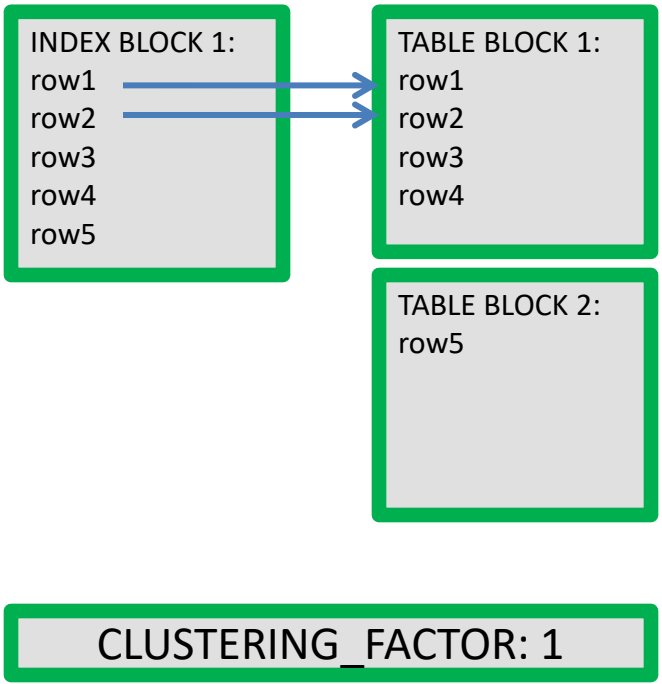
CLUSTERING FACTOR:



WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

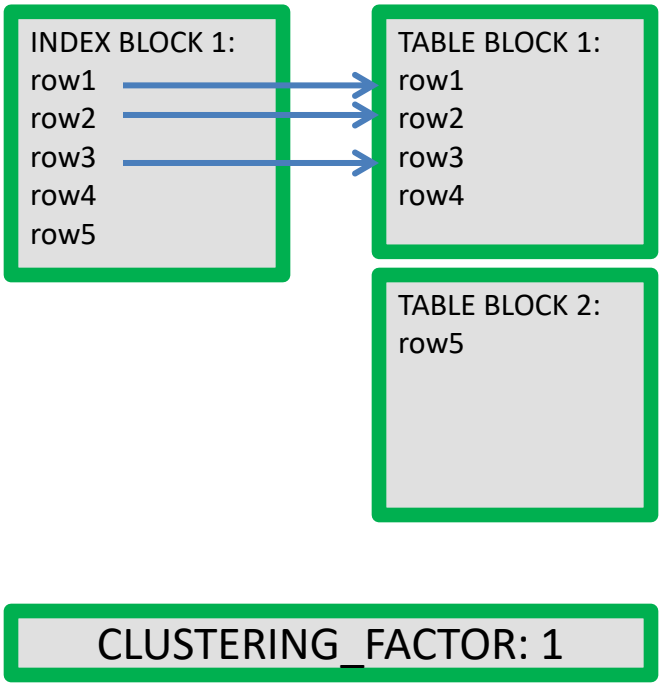
CLUSTERING FACTOR:



WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

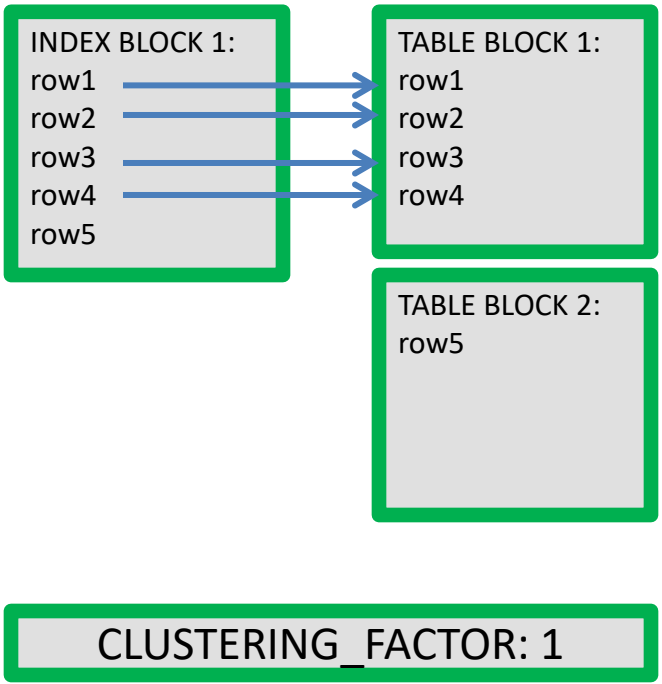
CLUSTERING FACTOR:



WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

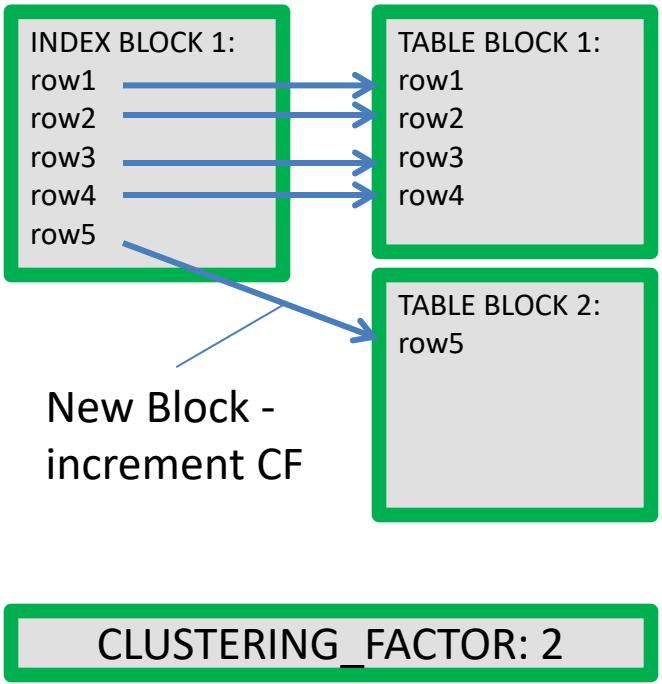
CLUSTERING FACTOR:



WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

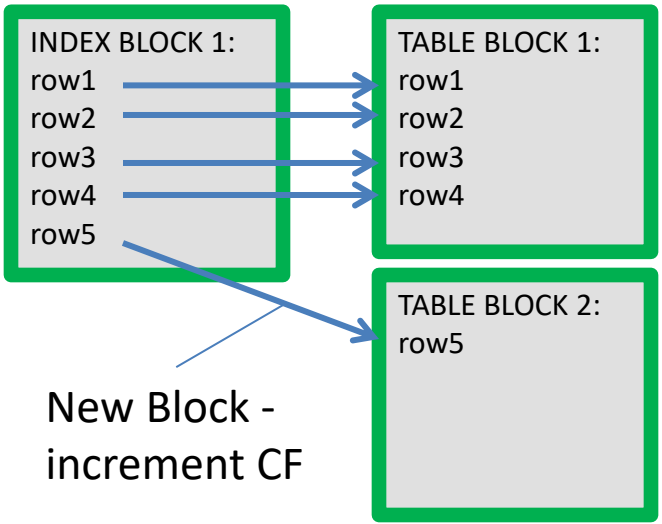
CLUSTERING FACTOR: 2



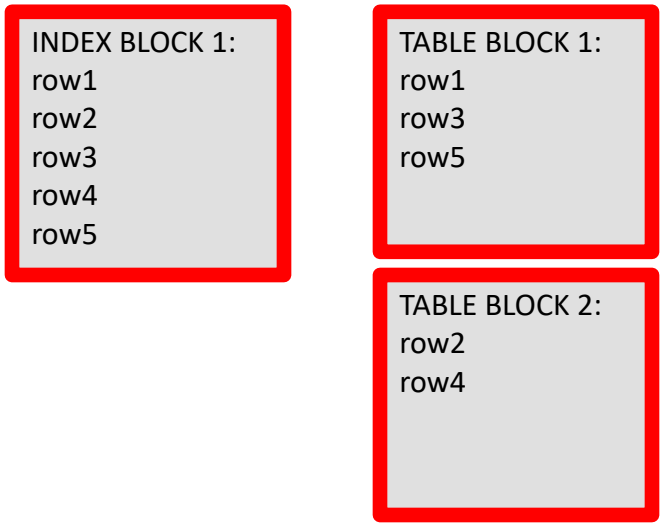
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:



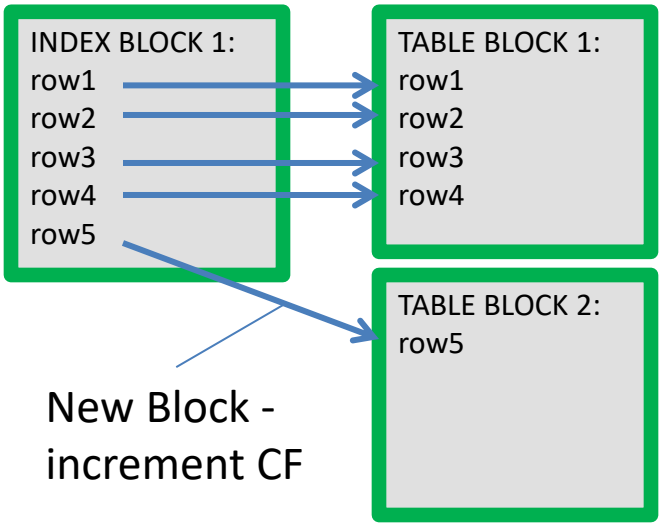
CLUSTERING_FACTOR: 2



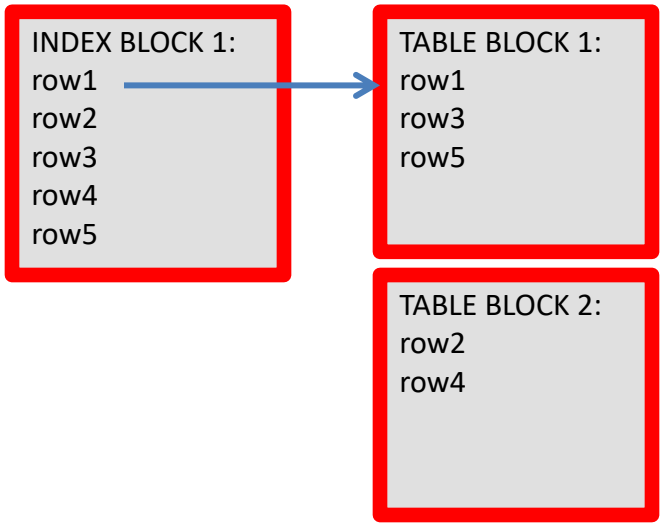
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:



CLUSTERING_FACTOR: 2

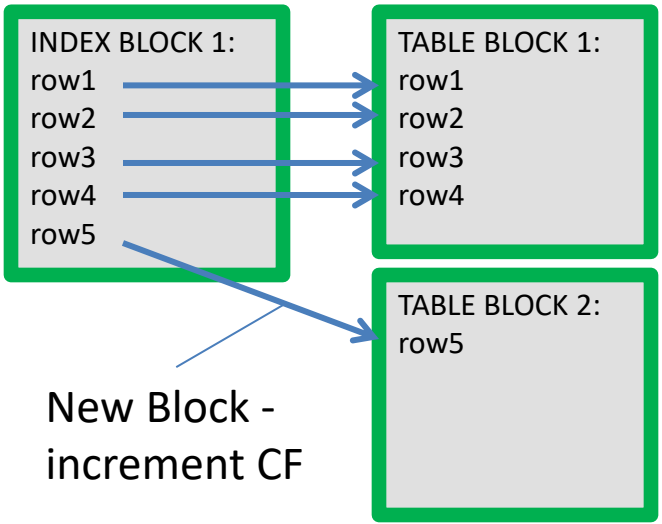


CLUSTERING_FACTOR: 1

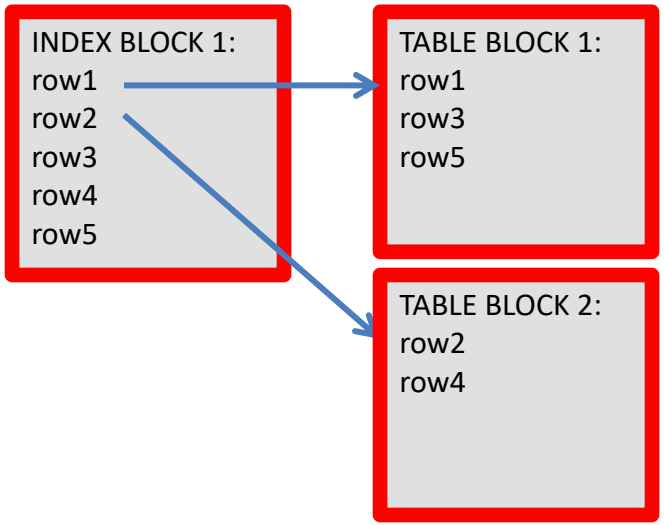
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:



CLUSTERING_FACTOR: 2

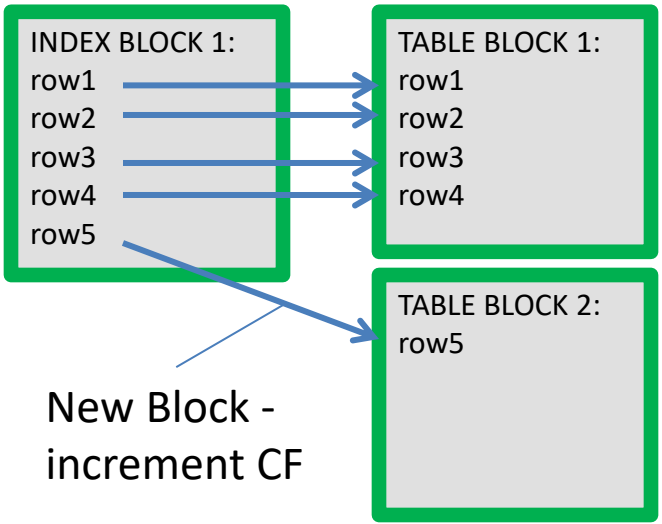


CLUSTERING_FACTOR: 2

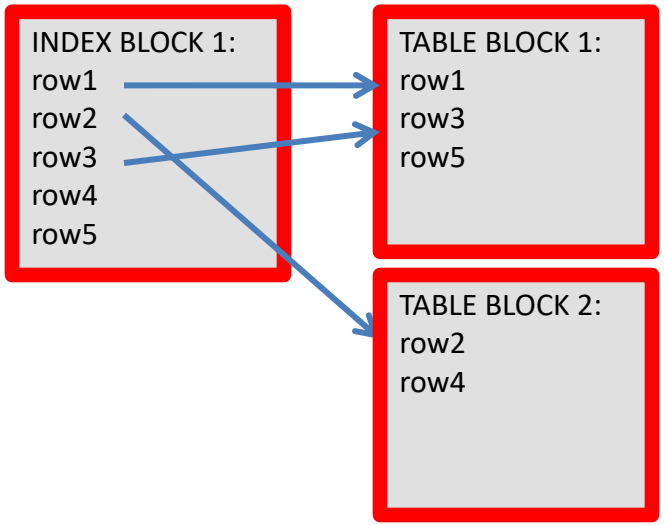
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:



CLUSTERING_FACTOR: 2

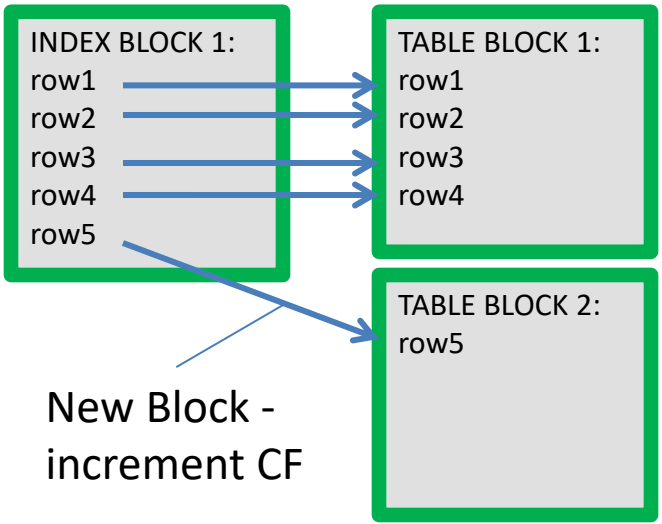


CLUSTERING_FACTOR: 3

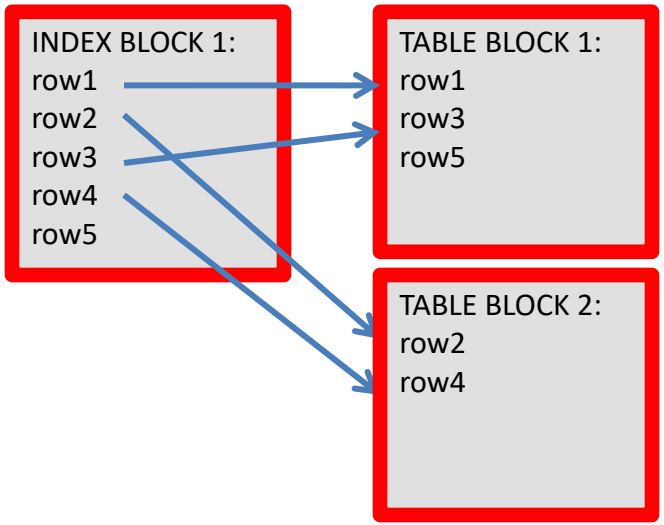
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:



CLUSTERING_FACTOR: 2

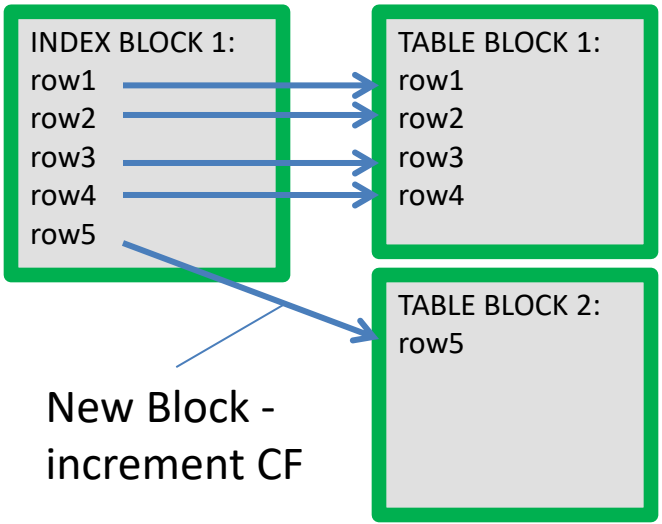


CLUSTERING_FACTOR: 4

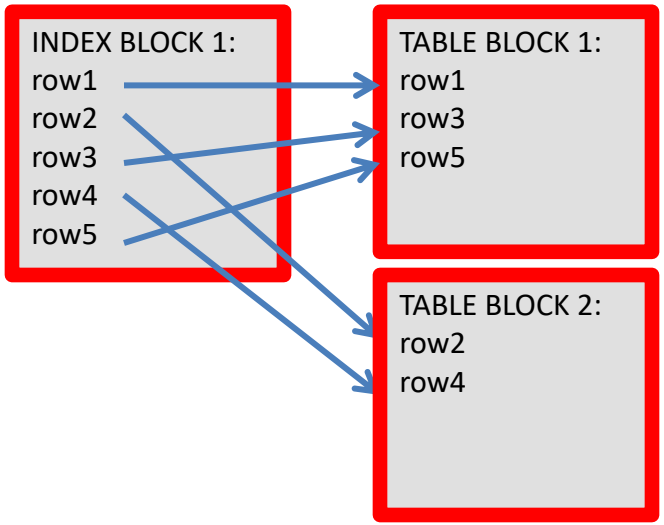
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

CLUSTERING FACTOR:



CLUSTERING_FACTOR: 2



CLUSTERING_FACTOR: 5

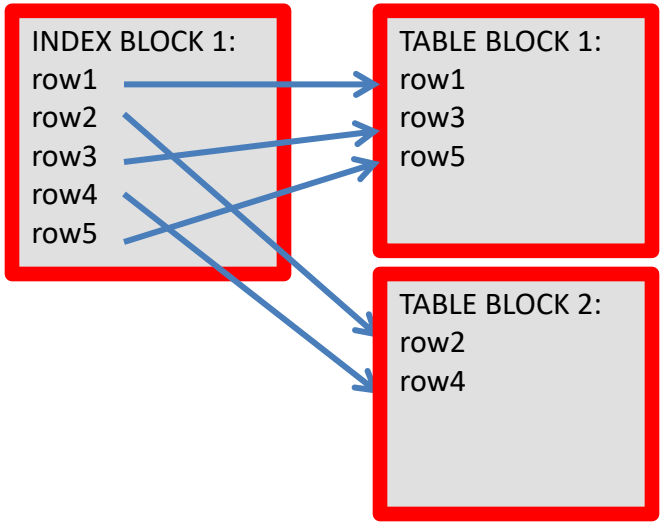
WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

Increase setting of TABLE_CACHED_BLOCKS so table blocks are assumed to be cached

DBMS_STATS.SET_GLOBAL_PREFS ('**TABLE_CACHED_BLOCKS**',16);

CLUSTERING_FACTOR: **2**



CLUSTERING_FACTOR: 5

WHAT ARE THESE STATISTICS ANYWAY?

DBA_IND_STATISTICS

NUM_ROWS	The number of rows in the index **= SCALE**
LEAF_BLOCKS	The number of leaf blocks in the index **= I/O**
DISTINCT_KEYS	Derive % of index to be used (NUM_ROWS/DISTINCT_KEYS)
CLUSTERING_FACTOR	How aligned is the index to the **table?**

As important to the Optimizer as index size!

If **CF** is close to #table ***blocks*** = **cheap index**

If **CF** is close to #table ***rows*** = **expensive index**

Rebuilding an index does not change the CF

WHAT ARE THESE STATISTICS ANYWAY?

DBA_**IND**_STATISTICS

NUM_ROWS	The number of rows in the index **= SCALE**
LEAF_BLOCKS	The number of leaf blocks in the index **= I/O**
DISTINCT_KEYS	Derive % of index to be used (NUM_ROWS/DISTINCT_KEYS)
CLUSTERING_FACTOR	How aligned is the index to the **table?**

As important to the Optimizer as index size!

If **CF** is close to #table ***blocks*** = **cheap index**

If **CF** is close to #table ***rows*** = **expensive index**

Rebuilding an index does not change the CF

From 12.1 **TABLE_CACHED_BLOCKS** stats gather option makes this number more realistic [or 11G PatchID is 15830250]

```
exec DBMS_STATS.SET_GLOBAL_PREFS('TABLE_CACHED_BLOCKS',16);
```

(a good starting point is 16 * #nodes in your cluster)

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_**HISTOGRAMS**

Histograms are additional statistics which try to describe the data distributions in a column

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_**HISTOGRAMS**

Histograms are additional statistics which try to describe the data distributions in a column

FREQUENCY	Super-Accurate and cheap to gather *from 12C* Will appear on every column with < 254 distinct values [NDV] that is used in a predicate (WHERE clause)

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_**HISTOGRAMS**

Histograms are additional statistics which try to describe the data distributions in a column

FREQUENCY	Super-Accurate and cheap to gather *from 12C* Will appear on every column with < 254 distinct values [NDV] that is used in a predicate (WHERE clause)
TOP-FREQUENCY	Mostly Accurate. Treat as FREQUENCY

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_**HISTOGRAMS**

Histograms are additional statistics which try to describe the data distributions in a column

FREQUENCY	Super-Accurate and cheap to gather *from 12C* Will appear on <u>every</u> column with < 254 distinct values [NDV] that is used in a predicate (WHERE clause)
TOP-FREQUENCY	Mostly Accurate. Treat as FREQUENCY
HEIGHT-BALANCED	replaced if you are using 12C and ESTIMATE_PERCENT is "AUTO_SAMPLE_SIZE" (default). Mostly useless and can cause plan stability issues.

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_**HISTOGRAMS**

Histograms are additional statistics which try to describe the data distributions in a column

FREQUENCY	Super-Accurate and cheap to gather *from 12C* Will appear on <u>every</u> column with < 254 distinct values [NDV] that is used in a predicate (WHERE clause)
TOP-FREQUENCY	Mostly Accurate. Treat as FREQUENCY
HEIGHT-BALANCED	replaced if you are using 12C and ESTIMATE_PERCENT is "AUTO_SAMPLE_SIZE" (default). Mostly useless and can cause plan stability issues.
HYBRID	Replaces H-B, are **sampled** [can cause plan stability issues] BUT a notable improvement *Should* only be created on skewed data sets with > 254 NDV that appear in predicates

Pre-12C, there are only (sampled) FREQUENCY and HEIGHT-BALANCED histograms.

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

ENDPOINT_NUMBER	How many values since the last endpoint?
ENDPOINT_ACTUAL_VALUE	The data value in the table
ENDPOINT_REPEAT_COUNT	For Hybrid Histograms - tells you how many values there are for the entry, making the it more like a Frequency Histogram for entries captured in the statistics.

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9

HISTOGRAM: NONE!

COLUMN	ENDPOINT_NUMBER	ENDPOINT_VALUE
CHANNEL_ID	0	2
CHANNEL_ID	1	9

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999

HISTOGRAM: FREQUENCY

COLUMN	ENDPOINT_NUMBER	VALUE	Maths
	0		
PROMO_ID	2074	33	2074- 0 = 2074 of "33"

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999

HISTOGRAM: FREQUENCY

COLUMN	ENDPOINT_NUMBER	VALUE	Maths
	0		
PROMO_ID	2074	33	2074- 0 = 2074 of "33"
PROMO_ID	20096	350	20096- 2074 = 18022 of "350"

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999

HISTOGRAM: FREQUENCY

COLUMN	ENDPOINT_NUMBER	VALUE	Maths
	0		
PROMO_ID	2074	33	2074- 0 = 2074 of "33"
PROMO_ID	20096	350	20096- 2074 = 18022 of "350"
PROMO_ID	31006	351	31006-20096 = 10910 of "351"
PROMO_ID	918843	999	918843-31006 = 887837 of "999"

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999

HISTOGRAM: FREQUENCY

COLUMN	ENDPOINT_NUMBER	VALUE	Maths	
	0			
PROMO_ID	2074	33	2074- 0 = 2074 of "33"	
PROMO_ID	20096	350	20096- 2074 = 18022 of "350"	1% - index?
PROMO_ID	31006	351	31006-20096 = 10910 of "351"	
PROMO_ID	918843	999	918843-31006 = 887837 of "999"	96% - FTS

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999
SALES	AMOUNT_SOLD	HYBRID	254	C10729	C2125349	6.4	1782.72

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999
SALES	AMOUNT_SOLD	HYBRID	254	C10729	C2125349	6.4	1782.72

HISTOGRAM: HYBRID

COLUMN	ENDP_#	VALUE	ENDPOINT_REPEAT_COUNT	Maths
AMOUNT_SOLD	1	6.40	9	1 - 0
AMOUNT_SOLD	33	7.10	19	33 - 1 = 32

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999
SALES	AMOUNT_SOLD	HYBRID	254	C10729	C2125349	6.4	1782.72

HISTOGRAM: HYBRID

COLUMN	ENDP_#	VALUE	ENDPOINT_REPEAT_COUNT	Maths	
AMOUNT_SOLD	1	6.40	9	~~1 - 0~~	**9** of "6.40"
AMOUNT_SOLD	33	7.10	19	~~33 - 1 - 32~~	**19** of "7.10"

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	9
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	999
SALES	AMOUNT_SOLD	HYBRID	254	C10729	C2125349	6.4	1782.72

HISTOGRAM: HYBRID

COLUMN	ENDP_#	VALUE	ENDPOINT_REPEAT_COUNT	Maths	
AMOUNT_SOLD	1	6.40	9	1 − 0	9 of "6.40"
AMOUNT_SOLD	33	7.10	19	33 − 1 − 32	19 of "7.10"
AMOUNT_SOLD	56	7.31	4	56 − 33 − 23	4 of "7.31"
AMOUNT_SOLD	78	7.44	3	78 − 56 − 22	3 of "7.44"

…etc…

WHAT ARE THESE STATISTICS ANYWAY?

DBA_TAB_HISTOGRAMS

DBA_TAB_COLUMNS

Table	Column	Histogram	#Buckets	Low_Val	High_Val	Decode_Low	Decode_High
SALES	CHANNEL_ID	NONE	1	C103	C10A	2	
SALES	PROMO_ID	FREQUENCY	4	C122	C20A64	33	
SALES	AMOUNT_SOLD	HYBRID	254	C10729	C2125349	6.4	1782.72

scale up by
num_rows/sample_size
to get #rows for the value

HISTOGRAM: HYBRID

COLUMN	ENDP_#	VALUE	ENDPOINT_REPEAT_COUNT	Maths	
AMOUNT_SOLD	1	6.40	9	1 - 0 =	9 of "6.40"
AMOUNT_SOLD	33	7.10	19	33 - 1 = 32	19 of "7.10"
AMOUNT_SOLD	56	7.31	4	56 - 33 = 23	4 of "7.31"
AMOUNT_SOLD	78	7.44	3	78 - 56 = 22	3 of "7.44"

…etc…

Oracle "Internal" Statistics

Oracle "Internal" Statistics

Dictionary Stats

Oracle 12+ gathers dictionary stats automatically

- Gather them **yourself** when your schemas are in place

Oracle "Internal" Statistics

Dictionary Stats

Oracle 12+ gathers dictionary stats automatically

- Gather them **yourself** when your schemas are in place
- Re-gather
  - if you make significant schema change
  - before an upgrade
  - after an upgrade


```
exec DBMS_STATS.GATHER_DICTIONARY_STATS;
```

Oracle "Internal" Statistics

Fixed Object Stats (X$ tables)

Oracle 12+ gathers **missing** stats automatically at the **end** of the maintenance window *(if there's time)*

- Gather them **yourself** during a *representative* workload


```
SELECT *
FROM dba_tab_statistics
WHERE object_type = 'FIXED TABLE'
```

Oracle "Internal" Statistics

Fixed Object Stats (X$ tables)

Oracle 12+ gathers **missing** stats automatically at the **end** of the maintenance window *(if there's time)*

- Gather them **yourself** during a *representative* workload
- Re-gather if you make changes to instance structure, such as SGA size, or workload changes


```
exec DBMS_STATS.DELETE_FIXED_OBJECT_STATS;
exec DBMS_STATS.GATHER_FIXED_OBJECTS_STATS;
```

```
SELECT *
FROM dba_tab_statistics
WHERE object_type = 'FIXED TABLE'
```

Oracle "Internal" Statistics

System Statistics

Measures your CPU and storage capabilities

Should we be gathering systems statistics?

Oracle "Internal" Statistics

System Statistics

Measures your CPU and storage capabilities

Should we be gathering systems statistics?

F**K NO!

(other opinions are available. They are wrong.)

**unless you are using a dedicated Exadata for a true Data Warehouse and you really want lots of "offloaded" table scans:

```
exec dbms_stats.gather_system_stats('EXADATA');
```

YOUR STATISTICS

The Gathering

Automatic Statistics Gathering

Automatic Statistics Gathering

DBA_AUTOTASK_CLIENT : auto optimizer stats collection
Gathers for every object that is STALE - DEFAULT 10% changed

And it runs: **DBA_AUTOTASK_SCHEDULE**

Window	Start Time	Duration Allowed
MONDAY_WINDOW	22.00.00	+00 04:00:00.000000
TUESDAY_WINDOW	22.00.00	+00 04:00:00.000000
WEDNESDAY_WINDOW	22.00.00	+00 04:00:00.000000
THURSDAY_WINDOW	22.00.00	+00 04:00:00.000000
FRIDAY_WINDOW	22.00.00	+00 04:00:00.000000
SATURDAY_WINDOW	06.00.00	+00 20:00:00.000000
SUNDAY_WINDOW	06.00.00	+00 20:00:00.000000

YOUR STATISTICS

Automatic Statistics Gathering

DBA_AUTOTASK_CLIENT auto optimizer stats collection

Gathers for every object that is STALE - DEFAULT 10% changed

And it runs in DBA_AUTOTASK_SCHEDULE

Timing is IMPORTANT!

Is your data representative at 22:00? Would 10:00 be better?

Is your system idle ALL WEEKEND?

Window	Next Time	Duration Allowed
MONDAY_WINDOW	22.00.00	+00 04:00:00.000000
TUESDAY_WINDOW	22.00.00	+00 04:00:00.000000
WEDNESDAY_WINDOW	22.00.00	+00 04:00:00.000000
THURSDAY_WINDOW	22.00.00	+00 04:00:00.000000
FRIDAY_WINDOW	22.00.00	+00 04:00:00.000000
SATURDAY_WINDOW	06.00.00	+00 20:00:00.000000
SUNDAY_WINDOW	06.00.00	+00 20:00:00.000000

YOUR STATISTICS

Automatic Statistics Gathering

DBA_AUTOTASK_CLIENT - auto optimizer stats collection

Gathers for every object that is STALE - DEFAULT 10% changed

And it runs in DBA_AUTOTASK_SCHEDULE

Window	Next Time	Duration Allowed
MONDAY_WINDOW	22.00.00	+00 04:00:00.000000
TUESDAY_WINDOW	22.00.00	+00 04:00:00.000000
WEDNESDAY_WINDOW	22.00.00	+00 04:00:00.000000
THURSDAY_WINDOW	22.00.00	+00 04:00:00.000000
FRIDAY_WINDOW	22.00.00	+00 04:00:00.000000
SATURDAY_WINDOW	06.00.00	+00 20:00:00.000000
SUNDAY_WINDOW	06.00.00	+00 20:00:00.000000

Timing is IMPORTANT!

Is your data representative at 22:00? Would 10:00 be better?

Is your system idle ALL WEEKEND?

Duration is IMPORTANT!

Can you do your stats gather in 4 hours?

REAL WORLD

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data

REAL WORLD

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):

19:00 load data

20:00 `dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)`

REAL WORLD

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):

19:00 load data

20:00 `dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)`

21:00 perform data manipulation with lovely new stats

REAL WORLD

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):

19:00 load data

20:00 `dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)`

21:00 perform data manipulation with lovely new stats

22:00 Autotask job does the gather again (stale dependent), with different (default) options

REAL WORLD

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):

19:00 load data

20:00 `dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)`

21:00 perform data manipulation with lovely new stats

22:00 Autotask job does the gather again (stale dependent), with different (default) options

04:00 A manual job starts and gathers the stats again.

They had disabled the Autotask stats gather years ago… an upgrade re-enabled it

REAL WORLD

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):

19:00 load data

20:00 `dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)`

21:00 perform data manipulation with lovely new stats

22:00 Autotask job does the gather again (stale dependent), with different (default) options

04:00 A manual job starts and gathers the stats again.

They had disabled the Autotask stats gather years ago… an upgrade re-enabled it

Key Questions

1. How do we ensure it's always the same gather?
2. How do we avoid repeating work?
3. Do I have to re-write all of my batch?

REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):

19:00 load data

20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…

21:00 perform data manipulation with lovely new stats

22:00 Autotask job does the gather again (stale dependent), with different (defaul

04:00 A manual job starts and gathers the stats again.

1. How do we ensure it's always the same gather?

REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data
20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,...
21:00 perform data manipulation with lovely new stats
22:00 Autotask job does the gather again (stale dependent), with different (defaul
04:00 A manual job starts and gathers the stats again.

1. How do we ensure it's always the same gather?

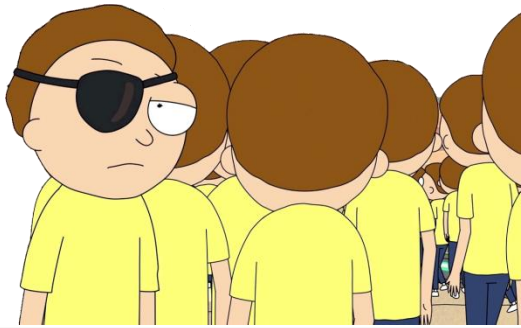
Don't specify options on the command line: use TABLE PREFS

```
DBMS_STATS.SET_TABLE_PREFS
(user,'TABLE_X',
'ESTIMATE_PERCENT',DBMS_STATS.AUTO_SAMPLE_SIZE)
```

REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data
20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)
21:00 perform data manipulation with lovely new stats
22:00 Autotask job does the gather again (stale dependent), with different (default) options
04:00 A manual job starts and gathers the stats again.

2. How do we avoid repeating work?



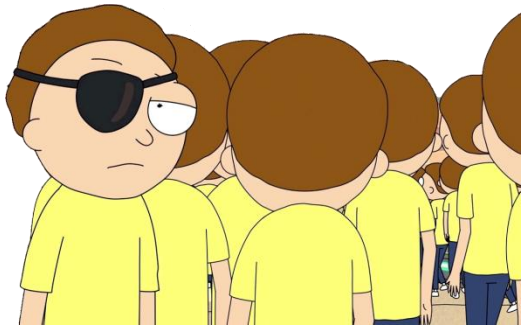
REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data
20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)
21:00 perform data manipulation with lovely new stats
22:00 Autotask job does the gather again (stale dependent), with different (default) options
04:00 A manual job starts and gathers the stats again.

2. How do we avoid repeating work?

Adjust Windows? `DBMS_SCHEDULER.set_attribute`

Disable Autotask & start it yourself? `DBMS_AUTO_TASK_IMMEDIATE.GATHER_OPTIMIZER_STATS`



REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data
20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)
21:00 perform data manipulation with lovely new stats
22:00 Autotask job does the gather again (stale dependent), with different (default) options
04:00 A manual job starts and gathers the stats again.

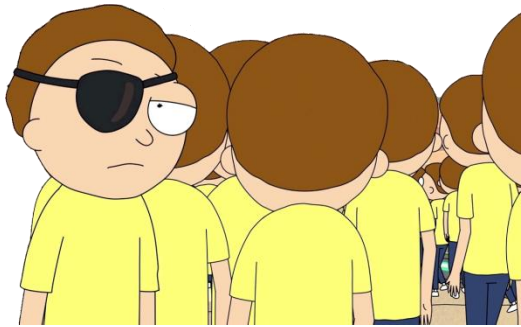
2. How do we avoid repeating work?

Adjust Windows? `DBMS_SCHEDULER.set_attribute`

Disable Autotask & start it yourself? `DBMS_AUTO_TASK_IMMEDIATE.GATHER_OPTIMIZER_STATS`

Maybe adjust the STALE percent for that table?

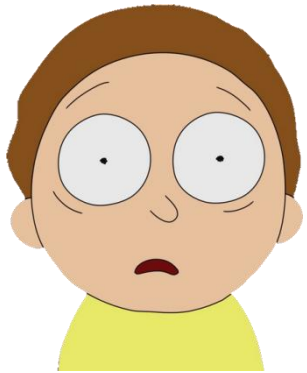
```
DBMS_STATS.SET_TABLE_PREFS
(user,'TABLE_X', 'STALE_PERCENT','50')
```



REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data
20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)
21:00 perform data manipulation with lovely new stats
22:00 Autotask job does the gather again (stale dependent), with different (default) options
04:00 A manual job starts and gathers the stats again.

3. Do I have to re-write lots of my code?

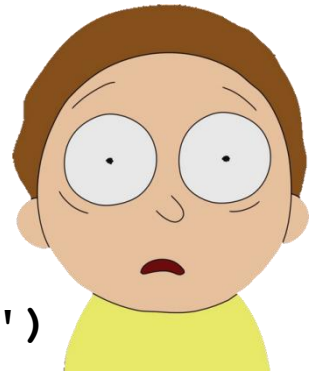


REAL-WORLD : TABLE_PREFS

BATCH CODE in 12.2 database (written *long ago* in Oracle 10G):
19:00 load data
20:00 dbms_stats.gather_table_stats(USER,'TABLE_X',ESTIMATE_PERCENT=>10,…)
21:00 perform data manipulation with lovely new stats
22:00 Autotask job does the gather again (stale dependent), with different (default) options
04:00 A manual job starts and gathers the stats again.

3. Do I have to re-write lots of my code?

From 12.2 we can OVERRIDE command-line options, forcing the stats gather to use TABLE PREFS



```
DBMS_STATS.SET_TABLE_PREFS
(user,'TABLE_X', 'PREFERENCE_OVERRIDES_PARAMETER','TRUE')
```

REAL-WORLD : TABLE_PREFS

SELECT * FROM **DBA_TAB_STAT_PREFS**

TABLE	PREFERENCE_NAME	PREFERENCE_VALUE
CUSTOMERS	METHOD_OPT	FOR ALL COLUMNS SIZE 1 FOR COLUMNS SIZE 1000 COUNTRY_ID,CUST_ID FOR COLUMNS SIZE AUTO CUST_GENDER
SALES	ESTIMATE_PERCENT	.000000
SALES	PREFERENCE_OVERRIDES_PARAMETER	TRUE

REAL-WORLD : GLOBAL_PREFS

You can also set preferences at a GLOBAL level:

```
exec DBMS_STATS.SET_GLOBAL_PREFS('METHOD_OPT','FOR ALL COLUMNS SIZE 1');
```

REAL-WORLD : GLOBAL_PREFS

You can also set preferences at a GLOBAL level:

```
exec DBMS_STATS.SET_GLOBAL_PREFS('METHOD_OPT','FOR ALL COLUMNS SIZE 1');
```

```
GLOBAL_PREFS in 19C (12 prefs in 11G, 14 new in 12C/18C/19C)
select dbms_stats.get_prefs('CASCADE') from dual;
select dbms_stats.get_prefs('DEGREE') from dual;
select dbms_stats.get_prefs('ESTIMATE_PERCENT') from dual;
select dbms_stats.get_prefs('METHOD_OPT') from dual;
select dbms_stats.get_prefs('NO_INVALIDATE') from dual;
select dbms_stats.get_prefs('GRANULARITY') from dual;
select dbms_stats.get_prefs('PUBLISH') from dual;
select dbms_stats.get_prefs('INCREMENTAL') from dual;
select dbms_stats.get_prefs('INCREMENTAL_LEVEL') from dual;
select dbms_stats.get_prefs('STALE_PERCENT') from dual;
select dbms_stats.get_prefs('AUTOSTATS_TARGET') from dual;
select dbms_stats.get_prefs('CONCURRENT') from dual;
select dbms_stats.get_prefs('INCREMENTAL_STALENESS') from dual;
select dbms_stats.get_prefs('GLOBAL_TEMP_TABLE_STATS') from dual;
select dbms_stats.get_prefs('TABLE_CACHED_BLOCKS') from dual;
select dbms_stats.get_prefs('OPTIONS') from dual;
select dbms_stats.get_prefs('STAT_CATEGORY') from dual;
select dbms_stats.get_prefs('PREFERENCE_OVERRIDES_PARAMETER') from dual;
select dbms_stats.get_prefs('APPROXIMATE_NDV_ALGORITHM') from dual;
select dbms_stats.get_prefs('AUTO_STAT_EXTENSIONS') from dual;
select dbms_stats.get_prefs('WAIT_TIME_TO_UPDATE_STATS') from dual;
select dbms_stats.get_prefs('ROOT_TRIGGER_PDB') from dual;
select dbms_stats.get_prefs('COORDINATOR_TRIGGER_SHARD') from dual;
select dbms_stats.get_prefs('AUTO_TASK_STATUS') from dual;
select dbms_stats.get_prefs('AUTO_TASK_MAX_RUN_TIME') from dual;
select dbms_stats.get_prefs('AUTO_TASK_INTERVAL') from dual;
```

STATISTICS NOT GATHERING

DBA_AUTOTASK_JOB_HISTORY

```
JOB_STATUS: STOPPED
JOB_INFO  : REASON="Stop job called because associated
                    window was closed"
```

STATISTICS NOT GATHERING

DBA_AUTOTASK_JOB_HISTORY

```
JOB_STATUS: STOPPED
JOB_INFO  : REASON="Stop job called because associated
                    window was closed"
```

Options: Make the window bigger

STATISTICS NOT GATHERING

DBA_AUTOTASK_JOB_HISTORY

```
JOB_STATUS: STOPPED
JOB_INFO  : REASON="Stop job called because associated
                    window was closed"
```

Options: Make the window bigger or

STATISTICS NOT GATHERING

DBA_AUTOTASK_JOB_HISTORY

```
JOB_STATUS: STOPPED
JOB_INFO  : REASON="Stop job called because associated
                    window was closed"
```

Options: Make the window bigger or **Speed It Up!!!**

STATISTICS NOT GATHERING

Speed It Up!!!

Parallelise the stats gather for a large table!

```
DBMS_STATS.SET_TABLE_PREFS
          ('SCHEMA', 'TABLE_X', 'DEGREE', 4)
```

STATISTICS NOT GATHERING

Speed It Up!!!

Parallelise the stats gather for a large table!

```
DBMS_STATS.SET_TABLE_PREFS
          ('SCHEMA', 'TABLE_X', 'DEGREE', 4)
```

You *could* be brave and let Oracle auto-parallelise all of it!

```
DBMS_STATS.SET_GLOBAL_PREFS
          ('DEGREE', DBMS_STATS.AUTO_DEGREE)
```

STATISTICS NOT GATHERING

Speed It Up!!!

Gather Statistics Concurrently - several tables at the same time…

DBMS_STATS.SET_GLOBAL_PREFS('**CONCURRENT**','**AUTOMATIC**')

MANUAL: Enabled only for manual statistics gathering
AUTOMATIC: Enabled only for the auto statistics gathering
ALL: Enabled for all statistics gathering calls
OFF: Concurrency is disabled (default)

Starts many Scheduler Jobs simultaneously!

Also need the following privileges:

```
CREATE JOB, MANAGE SCHEDULER, MANAGE ANY QUEUE
```

STATISTICS NOT GATHERING

Speed It Up!!!

Parallel or *Concurrent (or both)* gathering of stats means you should use **Resource Manager** to limit the total resources consumed

DBA_RSRC_CONSUMER_GROUPS: **ORA$AUTOTASK**

If you are doing ANY
PARALLEL processing
you ***really*** should use
Resource Manager
to control it!

STATISTICS NOT GATHERING

Resource Manager

```
begin
 dbms_resource_manager.clear_pending_area();
 dbms_resource_manager.create_pending_area();
 dbms_resource_manager.update_plan_directive(
  plan                          => 'DEFAULT_MAINTENANCE_PLAN',
  group_or_subplan              => 'ORA$AUTOTASK',
  new_mgmt_p1                   => 10,              -- Gets at least 10%
  new_max_utilization_limit     => 50,              -- Not allowed more than 50% (if the CPU is busy)
  new_parallel_degree_limit_p1 => 4);               -- And don't go wild…
 dbms_resource_manager.update_plan_directive(
  plan                          => 'DEFAULT_MAINTENANCE_PLAN',
  group_or_subplan              => 'OTHER_GROUPS',
  new_mgmt_p1                   => 20);
 dbms_resource_manager.update_plan_directive(
  plan                          => 'DEFAULT_MAINTENANCE_PLAN',
  group_or_subplan              => 'SYS_GROUP',
  new_mgmt_p1                   => 70);
  dbms_resource_manager.validate_pending_area;
  dbms_resource_manager.submit_pending_area;
end;
/
```

select … from DBA_RSRC_PLAN_DIRECTIVES
where plan = 'DEFAULT_MAINTENANCE_PLAN';

GROUP_OR_SUBPLAN	MGMT_P1	PARALLEL_DEGREE_LIMIT_P1	MAX_U
ORA$AUTOTASK	10	4	50
OTHER_GROUPS	20		
SYS_GROUP	70		

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

1. Set your stats
2. Lock the stats
3. Bask in the
   glow of
   consistent
   good plans?

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

1. Set your stats
2. Lock the stats
3. Bask in the glow of consistent good plans?

REAL WORLD

The client had an unusual "small" database which was only accessed via IOT's (i.e. no humans with unexpected inputs)

Got some good stats via calculations + (some) experiments

It was an unusual DB though. We also:

- disabled Hash Joins (NL was King here)

- NOARCHIVELOG mode

- Did NO backups

Locked it. Left it. Never gathered another statistic.

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

REAL WORLD EXAMPLE#2

This is actual code from a client. Did you set everything you need to set?

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

REAL WORLD EXAMPLE#2

This is actual code from a client. Did you set everything you need to set?

```
DBMS_STATS.SET_TABLE_STATS (
            ownname        => 'SCHEMA',
            tabname        => 'BIG_PART_TAB',
            numrows        => l_numrows,
            numblks        => l_num_blocks,
            avgrlen        => l_avgrow_len,
            no_invalidate => TRUE,
            force          => gather_force
            );
```

The input values for this are calculated from the (gathered) partitions.

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

REAL WORLD EXAMPLE#2

DBA_TAB_STATISTICS

NUM_ROWS	BLOCKS	AVG_ROW_LEN	SAMPLE_SIZE	LAST_ANALYZED	GLOBAL_STATS	USER_STATS
2,117,195,310	12,354,980	68	**587,940,830**	*(recently)*	YES	YES

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

REAL WORLD EXAMPLE#2

DBA_TAB_STATISTICS

NUM_ROWS	BLOCKS	AVG_ROW_LEN	SAMPLE_SIZE	LAST_ANALYZED	GLOBAL_STATS	USER_STATS
2,117,195,310	12,354,980	68	**587,940,830**	*(recently)*	YES	YES

DBA_TAB_COL_STATISTICS

COLUMN_NAME	NUM_DISTINCT	**LOW_VALUE**	**HIGH_VALUE**	LAST_ANALYZED	USER_STATS
UNIQUEish COLUMN	696,480	**10,002,011**	**919,120,110**	(a few years ago)	NO

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

REAL WORLD EXAMPLE#2

DBA_TAB_STATISTICS

NUM_ROWS	BLOCKS	AVG_ROW_LEN	SAMPLE_SIZE	LAST_ANALYZED	GLOBAL_STATS	USER_STATS
2,117,195,310	12,354,980	68	587,940,830	*(recently)*	YES	YES

DBA_TAB_COL_STATISTICS

COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	LAST_ANALYZED	USER_STATS
UNIQUEish COLUMN	696,480	10,002,011	919,120,110	(a few years ago)	NO

DBA_PART_COL_STATISTICS

COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	LAST_ANALYZED	USER_STATS
UNIQUEish COLUMN	256,150	10,002,011	4,620,540,110	(recently)	NO

MANUALLY SETTING STATISTICS

DBMS_STATS.SET_TABLE_STATS

REAL WORLD EXAMPLE#2

DBA_TAB_STATISTICS

NUM_ROWS	BLOCKS	AVG_ROW_LEN	SAMPLE_SIZE	LAST_ANALYZED	GLOBAL_STATS	USER_STATS
2,117,195,310	12,354,980	68	587,940,830	(recently)	YES	YES

DBA_TAB_COL_STATISTICS

COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	LAST_ANALYZED	USER_STATS
UNIQUEish COLUMN	69,480	10,002,011	919,120,110	(a few years ago)	NO

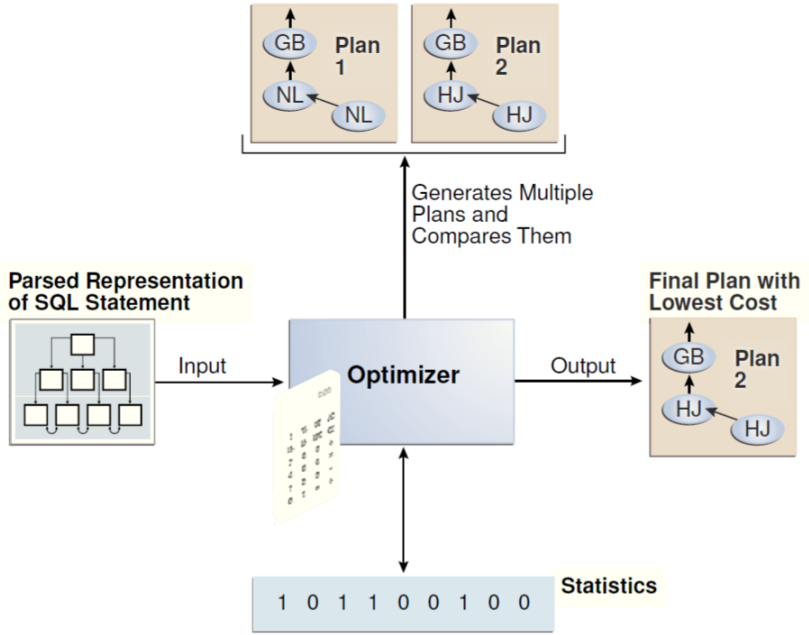
DBA_PART_COL_STATISTICS

COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	LAST_ANALYZED	USER_STATS
UNIQUEish COLUMN	256,150	10,002,011	4,620,540,110	(recently)	NO

Risk!
Did you set everything you need to set?
Did the bits you missed matter?

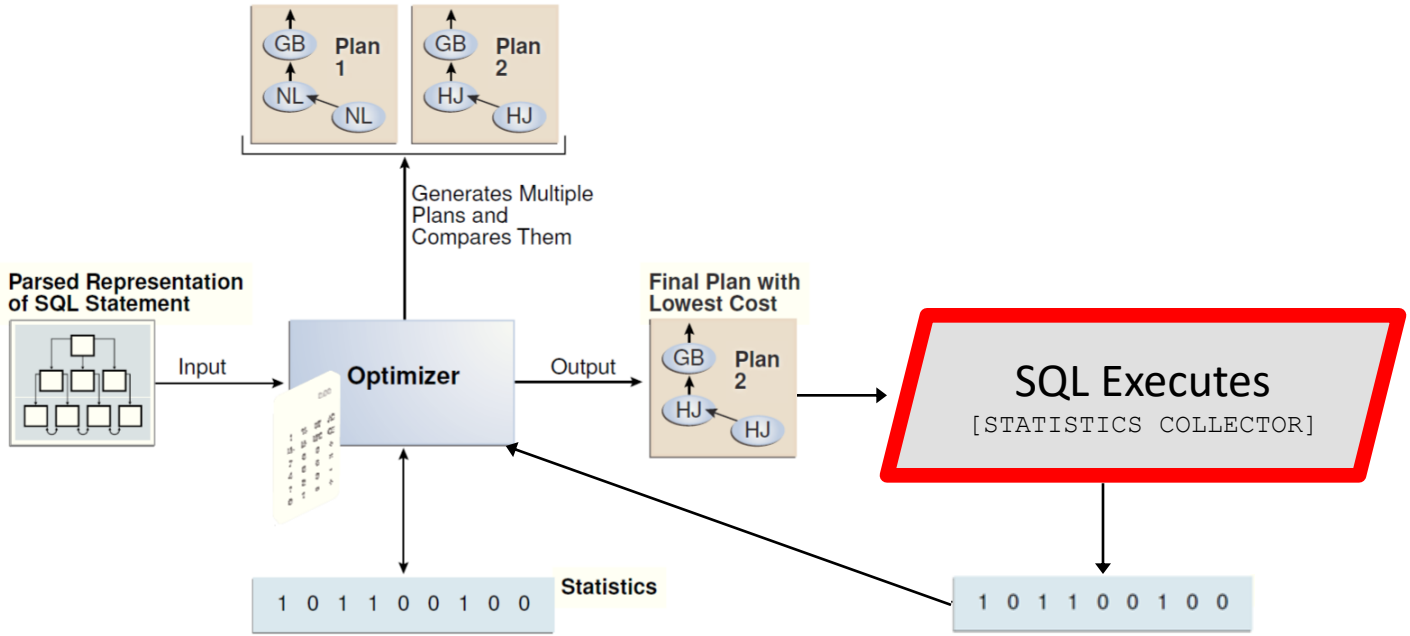
THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Statistical (*formerly Cardinality*) Feedback



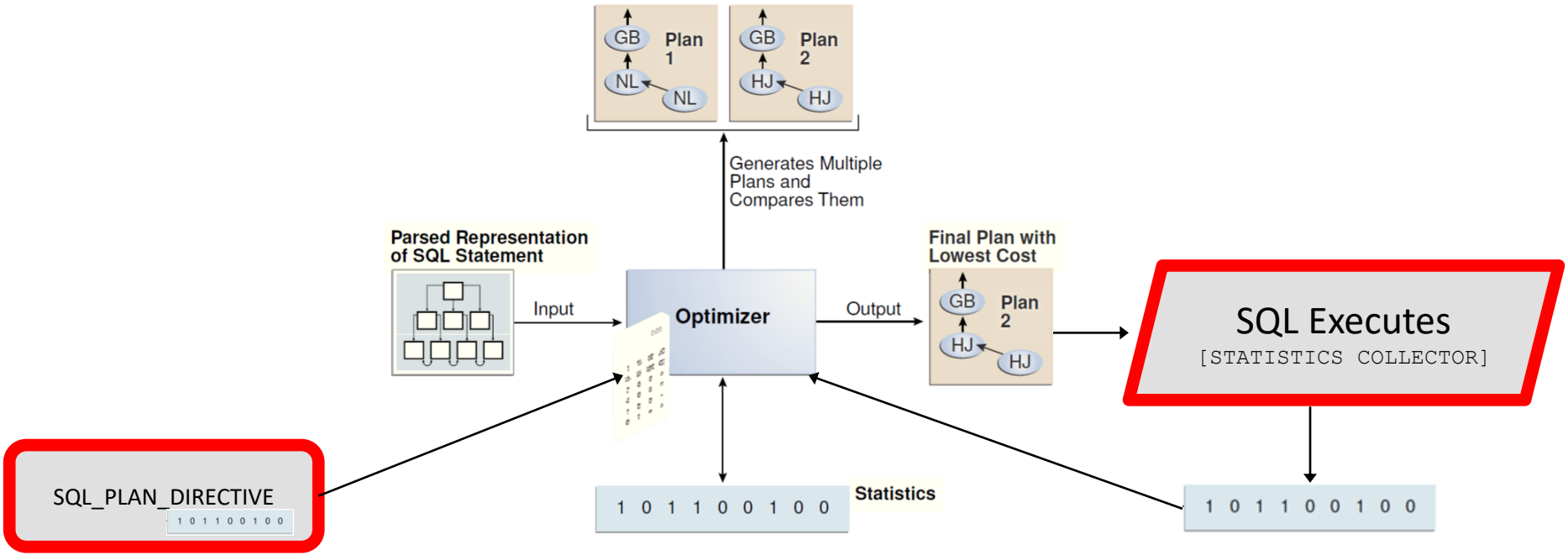
THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Statistical (*formerly Cardinality*) Feedback



THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Statistical (*formerly Cardinality*) Feedback



THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Statistical (*formerly Cardinality*) Feedback

You run some SQL, it may identify different statistical values:

The STATISTICS COLLECTOR results can be seen in: **`V$SQL_PLAN_STATISTICS_ALL`**

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Statistical (*formerly Cardinality*) Feedback

You run some SQL, it may identify different statistical values:

The STATISTICS COLLECTOR results can be seen in: **V$SQL_PLAN_STATISTICS_ALL**

OPERATION	OPTIONS	O-NAME	O-TYPE	CARDINALITY	OUTPUT ROWS
SELECT STATEMENT					2550
HASH JOIN				25	2550
NESTED LOOPS				25	5050
NESTED LOOPS				25	5050
STATISTICS COLLECTOR					5050
TABLE ACCESS	FULL	TAB2	TABLE	25	5050
INDEX	UNIQUE SCAN	TAB3_PK	INDEX (UNIQUE)	1	0
TABLE ACCESS	BY INDEX ROWID	TAB3	TABLE	1	0
TABLE ACCESS	FULL	TAB3	TABLE	1	7500

These statistics are only held in the SGA and will age out…

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Statistical (*formerly Cardinality*) Feedback

You run some SQL, it may identify different statistical values:

The STATISTICS COLLECTOR results can be seen in: **V$SQL_PLAN_STATISTICS_ALL**

OPERATION	OPTIONS	O-NAME	O-TYPE	CARDINALITY	OUTPUT ROWS
SELECT STATEMENT					2550
HASH JOIN				25	2550
NESTED LOOPS				25	5050
NESTED LOOPS				25	5050
STATISTICS COLLECTOR					5050
TABLE ACCESS	FULL	TAB2	TABLE	25	5050
INDEX	UNIQUE SCAN	TAB3_PK	INDEX (UNIQUE)	1	0
TABLE ACCESS	BY INDEX ROWID	TAB3	TABLE	1	0
TABLE ACCESS	FULL	TAB3	TABLE	1	7500

STATS expected 25 rows, we got 2550 & 5050

Feeds the new stats back into a re-Parse.
These statistics are only held in the SGA and will age out…

THOSE AREN'T THE STATS YOU'RE LOOKING FOR AT

Dynamic Statistics (Sampling) - `optimizer_dynamic_sampling`

Level	When does it use Dynamic Stats?
0	off
1	if No stats on an unpartitioned table, and no indexes, and table is bigger than 32 blocks (samples 32 blocks)
2 (default)	**if you have no stats on 1 table in the join *or*** *(and this is badly documented)* **if you use PARALLEL** (samples 64 blocks)
3	(as 2) + if you have a complex predicate expression [e.g WHERE SUBSTR(column,1,3) =]
4	(as 3) + an OR or AND between multiple predicates on the same table
5-10	(as 4) but sample 128/256/512/1024/4096/ALL table blocks
11	Automatically Determined by Oracle

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Dynamic Statistics (Sampling) - `optimizer_dynamic_sampling`

Level	When does it use Dynamic Stats?
0	off
1	if No stats on an unpartitioned table, and no indexes, and table is bigger than 32 blocks (samples 32 blocks)
2 (default)	**if you have no stats on 1 table in the join *or*** *(and this is badly documented)* **if you use PARALLEL** (samples 64 blocks)
3	(as 2) + if you have a complex predicate expression [e.g WHERE SUBSTR(column,1,3) =]
4	(as 3) + an OR or AND between multiple predicates on the same table
5-10	(as 4) but sample 128/256/512/1024/4096/ALL table blocks
11	Automatically Determined by Oracle

If you have a 10,000,000,000 row table, how representative is a 4096 block sample?
If you are looking for consistent plans (i.e. consistent stats for OLTP), should you use this?

THOSE AREN'T THE STATS YOU'RE LOOKING FOR AT

Dynamic Statistics (Sampling) - optimizer_dynamic_sampling

In the **Notes** section of DBMS_XPLAN execution plan output it will show the level used:

```
Note
-----
 - dynamic statistics used: dynamic sampling  (level=7)   or
 - dynamic statistics used for this statement (level=4)

The level will vary depending upon data size if using PARALLEL query from level 2.
```

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Dynamic Statistics (Sampling) - optimizer_dynamic_sampling

In the **Notes** section of DBMS_XPLAN execution plan output it will show the level used:

```
Note
-----
 - dynamic statistics used: dynamic sampling  (level=7)   or
 - dynamic statistics used for this statement (level=4)

The level will vary depending upon data size if using PARALLEL query from level 2.
```

In V$SQL.SQL_TEXT you will see the dynamic sampling which is being executed:

Where your stats aren't good enough or are missing:

```
SELECT /* OPT_DYN_SAMP */ … SAMPLE BLOCK (0.51390, 8) SEED(1) "TABLE_X"
```

Where Dynamic Sampling/SQL Plan Directive has kicked in:

```
SELECT /* DS_SVC */       … SAMPLE BLOCK (0.51398, 8) SEED(1) "TABLE_X"
```

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

Adaptive Statistics

THOSE AREN'T THE STATS YOU'RE LOOKING FOR AT

Adaptive Statistics

12.1 Introduced and **Enabled** this by default
12.2 Retained the functionality but **Disabled** this feature by default
optimizer_adaptive_statistics=FALSE

What it does?

- Identify one or more COLUMNS with poor statistics (via Statistical Feedback)
- Create a SQL_PLAN_DIRECTIVE to perform Dynamic Sampling against those table column(s)
  (12.2+ also has a SQL_PLAN_DIRECTIVE to store Dynamically Sampled Stats)

To control in 12.1, see
"Recommendations for Adaptive Features in Oracle Database 12c Release 1 (12.1): (Doc ID 2187449.1)"

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

SQL Plan Directives

`DYNAMIC_SAMPLING:`
Instructs the optimizer to get "better" stats, now

`DYNAMIC_SAMPLING_RESULT:`
(12.2+) Stores the dynamically sampled statistics. Goes STALE like your normal stats and then you do another Dynamic Sample.

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

SQL Plan Directives

DYNAMIC_SAMPLING:
Instructs the optimizer to get "better" stats, now

DYNAMIC_SAMPLING_RESULT:
(12.2+) Stores the dynamically sampled statistics. Goes STALE like your normal stats and then you do another Dynamic Sample.

```
SELECT
    dspd.type,
    dspd.reason,
    dspdo.owner,
    dspdo.object_name,
    dspdo.subobject_name,
    dspdo.object_type
FROM
    dba_sql_plan_directives  dspd,
    dba_sql_plan_dir_objects dspdo
WHERE
    dspd.directive_id = dspdo.directive_id;
```

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

SQL Plan Directives

DYNAMIC_SAMPLING:
Instructs the optimizer to get "better" stats, now

DYNAMIC_SAMPLING_RESULT:
(12.2+) Stores the dynamically sampled statistics. Goes STALE like your normal stats and then you do another Dynamic Sample.

```
SELECT
    dspd.type,
    dspd.reason,
    dspdo.owner,
    dspdo.object_name,
    dspdo.subobject_name,
    dspdo.object_type
FROM
    dba_sql_plan_directives  dspd,
    dba_sql_plan_dir_objects dspdo
WHERE
    dspd.directive_id = dspdo.directive_id;
```

TYPE	REASON	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_TYPE
DYNAMIC_SAMPLING	SINGLE TABLE CARDINALITY MISESTIMATE	CUSTOMERS	CUST_CITY	COLUMN
DYNAMIC_SAMPLING	SINGLE TABLE CARDINALITY MISESTIMATE	CUSTOMERS	CUST_STATE_PROVINCE	COLUMN
DYNAMIC_SAMPLING	SINGLE TABLE CARDINALITY MISESTIMATE	CUSTOMERS		TABLE
DYNAMIC_SAMPLING_RESULT	VERIFY CARDINALITY ESTIMATE	CUSTOMERS	CUST_CITY	TABLE

THOSE AREN'T THE STATS YOU'RE LOOKING ~~FOR~~ AT

SQL Plan Directives

DYNAMIC_SAMPLING:
Instructs the optimizer to get "better" stats, now.

DYNAMIC_SAMPLING_RESULT:
(12.2+) Stores the dynamically sampled statistics.
Goes STALE like your normal stats and when you do another Dynamic Sample.

Dynamic Sampling can be bad for OLTP
Adaptive Statistics can be bad for OLTP

They ***can*** be of benefit in a true Data Warehouse/BI environment where you value the BEST execution plan and are prepared to spend time working it out.
BUT your system is different! TEST!

```
SELECT
       dspd.type,
       dspd.reason,
       dspdo.owner,
       dspdo.object_name,
       dspdo.subobject_name
       dspdo.object_type
FROM   dba_sql_plan_directives  dspd,
       dba_sql_plan_dir_objects dspdo
WHERE  dspd.directive_id = dspdo.directive_id;
```

TYPE	REASON	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_TYPE
DYNAMIC_SAMPLING	SINGLE TABLE CARDINALITY MISESTIMATE	CUSTOMERS	CUST_CITY	COLUMN
DYNAMIC_SAMPLING	SINGLE TABLE CARDINALITY MISESTIMATE	CUSTOMERS	CUST_STATE_PROVINCE	COLUMN
DYNAMIC_SAMPLING	SINGLE TABLE CARDINALITY MISESTIMATE	CUSTOMERS		TABLE
DYNAMIC_SAMPLING_RESULT	VERIFY CARDINALITY ESTIMATE	CUSTOMERS	CUST_CITY	TABLE

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
DECLARE
    task_name   VARCHAR2(128) := 'stats_advisor_report_task';
    exec_name   VARCHAR2(128) := NULL;
    report      CLOB := NULL;
BEGIN
  -- create a task
    task_name := dbms_stats.create_advisor_task(task_name);
  -- execute the task
    exec_name := dbms_stats.execute_advisor_task(task_name);
  -- view the task report
    report    := dbms_stats.report_advisor_task(task_name);
    dbms_output.put_line(report);


END;
```

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
DECLARE
    task_name   VARCHAR2(128) := 'stats_advisor_report_task';
    exec_name   VARCHAR2(128) := NULL;
    report      CLOB := NULL;
BEGIN
  -- create a task
    task_name := dbms_stats.create_advisor_task(task_name);
  -- execute the task
    exec_name := dbms_stats.execute_advisor_task(task_name);
  -- view the task report
    report    := dbms_stats.report_advisor_task(task_name);
    dbms_output.put_line(report);
  -- for the Brave implement the recommendation from the task
  --implementation_result := dbms_stats.implement_advisor_task(tname);
END;
```

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
--------------------------------------------------------------------------------
FINDINGS
--------------------------------------------------------------------------------
Rule Name:         UseConcurrent
Rule Description: Use Concurrent preference for Statistics Collection
```

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
--------------------------------------------------------------------------------
FINDINGS
--------------------------------------------------------------------------------
Rule Name:          UseConcurrent
Rule Description:   Use Concurrent preference for Statistics Collection
Finding:            The CONCURRENT preference is not used.
Recommendation:     Set the CONCURRENT preference.
Example:            dbms_stats.set_global_prefs('CONCURRENT', 'ALL');
```

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
---------------------------------------------------------------------------
FINDINGS
---------------------------------------------------------------------------
Rule Name:          UseConcurrent
Rule Description:   Use Concurrent preference for Statistics Collection
Finding:            The CONCURRENT preference is not used.
Recommendation:     Set the CONCURRENT preference.
Example:            dbms_stats.set_global_prefs('CONCURRENT', 'ALL');
Rationale:          The system's condition satisfies the use of concurrent statistics
                    gathering. Using CONCURRENT increases the efficiency of statistics
                    gathering.
```

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
-------------------------------------------------------------------------------
FINDINGS
-------------------------------------------------------------------------------
Rule Name:         UseDefaultPreference
Rule Description:  Use Default Preference for Stats Collection
Finding:           Global preference METHOD_OPT is set to a non-default value
                   'FOR ALL COLUMNS SIZE 1'.
```

WHAT TO DO?

Statistics Advisor

- Oracle 12.2 comes with a statistics advisor to help you
- It runs nightly (window permitting): AUTO_STATS_ADVISOR_TASK or you can run it yourself

```
-------------------------------------------------------------------------------
FINDINGS
-------------------------------------------------------------------------------
Rule Name:          UseDefaultPreference
Rule Description:   Use Default Preference for Stats Collection
Finding:            Global preference METHOD_OPT is set to a non-default value
                    'FOR ALL COLUMNS SIZE 1'.
Recommendation:     Set the value of preference METHOD_OPT to 'FOR ALL COLUMNS SIZE AUTO'.
Example:            Setting preference cascade to default value:
                    dbms_stats.set_global_prefs('CASCADE', NULL);
Rationale:          METHOD_OPT controls the creation of histograms during statistics collection.
                    With the default value FOR ALL COLUMNS SIZE AUTO, Oracle determines which
                    columns require histograms and the number of buckets to use based on the usage
                    of columns in SQL statements and the number of distinct values. The default
                    value helps to create the necessary histograms with an adequate number of
                    buckets.
```

WHAT TO DO?

Statistics Advisor: `V$STATS_ADVISOR_RULES;`

Rules can be filtered to avoid repeatedly reporting against specific settings you have made:
DBMS_STATS.CONFIGURE_ADVISOR_FILTER

WHAT TO DO?

Statistics Advisor: V$STATS_ADVISOR_RULES;

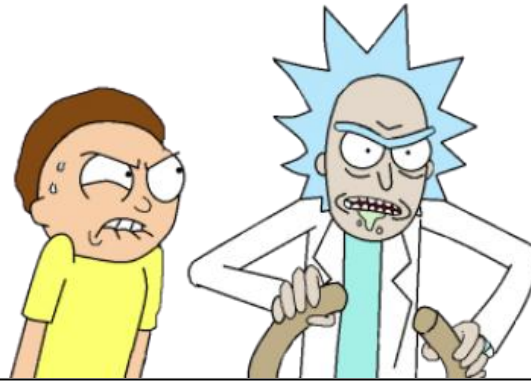
ID	NAME	RULE_TYPE	DESCRIPTION
1	UseAutoJob	SYSTEM	Use Auto Job for Statistics Collection
2	CompleteAutoJob	SYSTEM	Auto Statistics Gather Job should complete successfully
3	MaintainStatsHistory	SYSTEM	Maintain Statistics History
4	UseConcurrent	SYSTEM	Use Concurrent preference for Statistics Collection
5	UseDefaultPreference	SYSTEM	Use Default Preference for Stats Collection
6	TurnOnSQLPlanDirective	SYSTEM	SQL Plan Directives should not be disabled
7	AvoidSetProcedures	OPERATION	Avoid Set Statistics Procedures
8	UseDefaultParams	OPERATION	Use Default Parameters in Statistics Collection Procedures
9	UseGatherSchemaStats	OPERATION	Use gather_schema_stats procedure
10	AvoidInefficientStatsOprSeq	OPERATION	Avoid inefficient statistics operation sequences
11	AvoidUnnecessaryStatsCollection	OBJECT	Avoid unnecessary statistics collection
12	AvoidStaleStats	OBJECT	Avoid objects with stale or no statistics
…			
23	AvoidAnalyzeTable	OBJECT	Avoid using analyze table commands for statistics collection

Rules can be filtered to avoid repeatedly reporting against specific settings you have made:
DBMS_STATS.CONFIGURE_ADVISOR_FILTER

2 New Statistics Features

Exadata and Cloud ONLY

- Real-Time Statistics
- High-Frequency Automatic Optimizer Statistics Collection



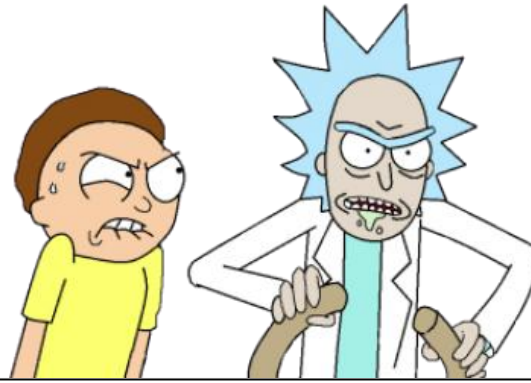
2 New Statistics Features

Exadata and Cloud ONLY

- Real-Time Statistics
- High-Frequency Automatic Optimizer Statistics Collection

Table 1-8 Performance

Feature / Option / Pack	SE2	EE	EE-ES	DBCS SE	DBCS EE	DBCS EE-HP	DBCS EE-EP	ExaCS	Notes
Real-Time Statistics	N	N	Y	Y	Y	Y	Y	Y	**EE-ES**: Available on Exadata. Not available on Oracle Database Appliance.
High-Frequency Automatic Optimizer Statistics Collection	N	N	Y	Y	Y	Y	Y	Y	**EE-ES**: Available on Exadata. Not available on Oracle Database Appliance.



Real-Time Statistics

- When a DML operation is currently modifying a table, the DB dynamically computes values for "the most essential statistics." (i.e. the easy ones)
  - LOW_VALUE
  - HIGH_VALUE
  - NUM_ROWS

Real-Time Statistics

- When a DML operation is currently modifying a table, the DB dynamically computes values for "the most essential statistics." (i.e. the easy ones)
  - LOW_VALUE
  - HIGH_VALUE
  - NUM_ROWS
- Reduces risk from "statistical decay & high-low value threat"

Real-Time Statistics

- When a DML operation is currently modifying a table, the DB dynamically computes values for "the most essential statistics." (i.e. the easy ones)
  - LOW_VALUE
  - HIGH_VALUE
  - NUM_ROWS
- Reduces risk from "statistical decay & high-low value threat"
- Does it during the DML into memory buffers and flushes to disk occasionally (DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO)

Real-Time Statistics

- When a DML operation is currently modifying a table, the DB dynamically computes values for "the most essential statistics." (i.e. the easy ones)
  - LOW_VALUE
  - HIGH_VALUE
  - NUM_ROWS
- Reduces risk from "statistical decay & high-low value threat"
- Does it during the DML into memory buffers and flushes to disk occasionally (DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO)
- Runs at an approximate 1% sample size to minimise impact to DML

Real-Time Statistics

- When a DML operation is currently modifying a table, the DB dynamically computes values for "the most essential statistics." (i.e. the easy ones)
  - LOW_VALUE
  - HIGH_VALUE
  - NUM_ROWS
- Reduces risk from "statistical decay & high-low value threat"
- Does it during the DML into memory buffers and flushes to disk occasionally (DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO)
- Runs at an approximate 1% sample size to minimise impact to DML
- Check DBA_TAB_STATISTICS.NOTES
  DBA_TAB_COL_STATISTICS.NOTES for value "**STATS_ON_CONVENTIONAL_DML**"

You still need to do traditional stats gathering!

High-Frequency Automatic Optimizer Statistics Collection

- Gathers "STALE" statistics much more frequently. Doesn't to any of the other autotask stats gathering stuff (e.g. Internal Stats, Stats Advisor)

19C

High-Frequency Automatic Optimizer Statistics Collection

- Gathers "STALE" statistics much more frequently. Doesn't to any of the other autotask stats gathering stuff (e.g. Internal Stats, Stats Advisor)

- Does not replace AUTOTASK stats gathering

19C

High-Frequency Automatic Optimizer Statistics Collection

- Gathers "STALE" statistics much more frequently. Doesn't to any of the other autotask stats gathering stuff (e.g. Internal Stats, Stats Advisor)
- Does not replace AUTOTASK stats gathering
- Will not run during the auto stats gather window

19C

High-Frequency Automatic Optimizer Statistics Collection

- Gathers "STALE" statistics much more frequently. Doesn't to any of the other autotask stats gathering stuff (e.g. Internal Stats, Stats Advisor)
- Does not replace AUTOTASK stats gathering
- Will not run during the auto stats gather window
- 3 new GLOBAL_PREFS
  - AUTO_TASK_STATUS ('OFF')
  - AUTO_TASK_MAX_RUN_TIME (3600 seconds)
  - AUTO_TASK_INTERVAL (run every 900 seconds. Min 60 seconds)

19c

High-Frequency Automatic Optimizer Statistics Collection

- Gathers "STALE" statistics much more frequently. Doesn't to any of the other autotask stats gathering stuff (e.g. Internal Stats, Stats Advisor)
- Does not replace AUTOTASK stats gathering
- Will not run during the auto stats gather window
- 3 new GLOBAL_PREFS
  - AUTO_TASK_STATUS ('OFF')
  - AUTO_TASK_MAX_RUN_TIME (3600 seconds)
  - AUTO_TASK_INTERVAL (run every 900 seconds. Min 60 seconds)
- Can see the runs in: DBA_AUTO_STAT_EXECUTIONS where ORIGIN='HIGH_FREQ_AUTO_TASK'

WHAT DIDN'T I COVER?

- Incremental Statistics for Partitioned Tables
- Synopses
- Partition Table Strategies using Copy
- Lots on Histograms
- Lots on Systems Statistics
- Statistics History & Rolling Back Stats to previous versions
- Pending Statistics to test plans before you unleash them
- Undocumented Parameters
- 10046 and 10053 traces
- and lots and lots of other stuff…

Time to use **The Force**

Questions?

BLOG: http://chandlerdba.com
Twitter: **@chandlerDBA**
E: neil@chandler.uk.com