



*Sybase Adaptive Server Enterprise*  
**Statistics Demystified**

---

Software Gems Pty Ltd  
Derek Ignatius Asirvadem  
Version 2.5  
17 Aug 20

# 1 Introduction

## 1.1 Purpose

Statistics are very important.

They form the basis upon which the Optimiser makes decisions when determining the Query Plan, and thus affect the performance of every query. The Optimiser in *Sybase ASE 15*<sup>1</sup> was a complete rewrite, to the inexperienced eye, it is more sensitive to Stats issues. More precisely, it uses Stats to make many more decisions than before. Therefore it is now more important than ever, to understand Stats, and to implement regular, and relevant, `UPDATE STATS` as an integral part of database maintenance.

### Purpose 1

The key criteria for Statistics are:

- they must be accurate, up-to-date, identifying the current content of the tables and columns
- they must be relevant:
  - maintaining Stats that are rarely used, or that do not change, lengthens the maintenance window
  - missing or stale Stats causes the Optimiser to make incorrect decisions, which typically affect performance
- they must be 'deep' enough (more, later) to be useful
- the decreasing size of the maintenance window in which Stats is to be updated

### Purpose 2

In general the *Sybase* manuals are poor, but in the area of Stats, they are abysmal<sup>2</sup>. This document intends to overcome that obstacle, in particular, it provides the information that is missing from the manuals. It answers these questions:

- What, in heaven's name, and precisely, is Statistics ?
- Why does every table need it ?
- How does one administer Statistics properly ?

### Purpose 3

The internet is a cesspool, full of misinformation, at best, it is superficial and fragmented information. In the *Sybase* community, the level of information is generally better, but we are bombarded with low level technical information that is irrelevant to the question being asked, leaving the seeker without the improved understanding that was sought, but more educated re trivia and more confused.

In the *Sybase* community the subject of Statistics, is the most mis-understood, particularly by the marketed "evangelists": they cannot teach what they do not know. So they double the volume of the bombardment to compensate for their insecurity, their inability to provide straight answers<sup>2</sup>.

If that confusion-without-resolution is familiar to you, you will need to erase everything you have heard about Statistics, and start again: this document presents the subject properly, logically, in increments. It may even make you immune to the bombardment of trivia that is made to sound important.

*The subject of Statistics is neither complicated, nor simple. The mechanics and considerations simply needs to be understood. However, it is made hugely complicated by the denizens of the web, particularly the self-styled "gurus". This document explains Statistics in a straight-forward manner, eliminating the complexity.*

This document is intended to assist any technically capable Administrator in covering all these requirements, and to do so efficiently.

## 1.2 Document Layout

The document is divided into the four subjects areas, plus one more for demonstration reports:

- |                                |                                                                                                                                                                                                                                                                                 |
|--------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2 <b>Concept</b>               | which every DBA should become thoroughly familiar with                                                                                                                                                                                                                          |
| 2.1 <b>Statistic Type</b>      | the different types, that need to be understood, and maintained                                                                                                                                                                                                                 |
| 2.2 <b>Column Stats</b>        | what it is; why it is important                                                                                                                                                                                                                                                 |
| 2.3 <b>Histogram</b>           | what it is; why it is important; its structure and content                                                                                                                                                                                                                      |
| 2.4 <b>ColumnGroup Stats</b>   | what it is; why it is important, indexed and not indexed                                                                                                                                                                                                                        |
| 2.5 <b>optdiag Stats</b>       | Stats reported by the utility                                                                                                                                                                                                                                                   |
| 2.6 <b>Summary</b>             | and ultimately, the imperative                                                                                                                                                                                                                                                  |
| 3 <b>Catalogue</b>             | required for anyone coding their own reports, to inspect and manage their Stats: explains the logical and physical tables; the rows; how the cells in the histogram are stored                                                                                                  |
| 4 <b>UPDATE STATS</b>          | a detailed explanation, and effect, of <b>each flavour</b> of the command, using a full, consolidated example, in a sequence                                                                                                                                                    |
| 5 <b>Goal</b>                  | to work towards, rather than simple instructions, in order to produce an efficient and relevant Update Stats, in the smallest maintenance window (Customers only)                                                                                                               |
| 6 <b>Demonstration Reports</b> | Software Gems' <i>HelpStatistic</i> and <i>HelpHistogram</i> reports, not for the purpose of marketing, but for detailed explanation; to identify what is important and not; how to structure your own reports; the logical grouping of information, to obtain maximum benefit. |

## 1.3 Document Status

Check this document for updates:

[Sybase Statistics Demystified](#)

Check all documents for updates:

[Software Gems' Sybase GEM Documents](#)

While this document provides information that is absent from the manuals, it is not intended to replace the product reference manuals; for full syntax and options, refer to the manuals.

<sup>1</sup> This document applies to *Sybase ASE 15.7*. Of course it applies to *ASE 16*, because it is upward compatible, but it does not define any New Features therein.

<sup>2</sup> The P&T/Stats manual is arguably the worst, since it presents very important information, but uses a schizophrenic style of expression; uses standard terms out of context; and uses terms without defining them, thus causing much confusion. Further, in the subject of *Sybase Statistics*, some writers do have good information but it is limited to a narrow area, and they are confused about the rest, the result being that confusion is spread far and wide. Thus a logical (full, clear, top-down) explanation is demanded.

## 2.1 Statistic Type

There are three **kinds of data** that may be loosely called *Statistics* that are stored in the ASE catalogue:

- **Density and Distribution Stats**  
for Columns & ColumnGroups are stored in `sysstatistics`; that is what is usually meant when the term *Statistics* is used, and that is the focus of this document. These are the table-level, content-based Stats that influence all query performance (that which the cost-based Optimiser uses when it makes its decisions) on the table
- **Index Metrics**  
are stored in `systabstats` (with two exceptions). Some of these values are updated only by `UPDATE STATISTICS` and `sp_flushstats`: that is therefore included in this document <sup>3</sup>
- **Query Metrics**  
historical processing metrics for individual queries are stored in `sysquerymetrics`; that is not included in this document.

In that first category, there are three **types of Statistics** that *Sybase can* capture:

For Each	Sybase Can Store	Also known as
ColumnGroup	ColumnGroup Stats	ColumnGroup Density
Column	Column Stats	Column Density
	Histogram	Distribution

Of course *Sybase ASE* actually captures only those Stats that you explicitly or implicitly request via the various flavours of `UPDATE STATS` command. That is where some of the confusion lies, and that is explained later.

## 2.2 Column Stats

Column Stats consist of two types.

### 2.2.1 Density

For each Column, *ASE* can store Density and Selectivity Stats.

```

Statistics for column:           "Component"
Last update of column statistics: Aug 11 2011 11:08:31:346AM

Range cell density:             0.0000000000000000
Total density:                  0.1025191733659713
Range selectivity:              default used (0.33)
In between selectivity:        default used (0.25)
Unique range values:           0.0000000000000000
Unique total values:           0.0833333333333333
Average column width:          default used (3.00)
    
```

*Where Distribution Stats are not explicitly captured, this stands as rudimentary Distribution Stats. This is what the poor end of town struggle with*

### 2.2.2 Distribution

Second, for each Column, *ASE* can store an Histogram, which contains full Distribution Stats.

```

Histogram for column:           "Component"
Column datatype:               char(3)
Requested step count:           20
Actual step count:             24
Sampling Percent:              10
Out of range Histogram Adjustment is ON.

Step    Weight    Value
-----
1       0.00000000    <    "Cac"
2       0.15133531    =    "Cac"
3       0.00000000    <    "Ctx"
4       0.08902077    =    "Ctx"
5       0.00000000    <    "Dsk"
6       0.04154303    =    "Dsk"
7       0.00000000    <    "Eng"
8       0.10089020    =    "Eng"
9       0.00000000    <    "Hkp"
10      0.08011869    =    "Hkp"
11      0.00000000    <    "Log"
12      0.13946587    =    "Log"
13      0.00000000    <    "Net"
14      0.05044510    =    "Net"
15      0.00000000    <    "Upr"
16      0.06231454    =    "Upr"
17      0.00000000    <    "Usy"
18      0.02670623    =    "Usy"
19      0.00000000    <    "Uth"
20      0.03560831    =    "Uth"
21      0.00000000    <    "Uvm"
22      0.09198813    =    "Uvm"
23      0.00000000    <    "Xct"
24      0.13056380    =    "Xct"
    
```

*More detail, later.*

*The output of the optdiag utility is used here, because most administrators are familiar with it. It is single-object oriented, or vertical; therefore, it is hard to navigate and follow, and voluminous.*

Column Stats are normally generated implicitly, due to either their participation in Indices, or being nominated as being frequently used [2.4]. In the event that Column Stats and an Histogram are desired for a Column that does not have it by implicit means, it can be generated explicitly, via `UPDATE STATS` [4.2]

<sup>3</sup> The *Sybase* utility `optdiag` reports two classes of information: Stats; and Index Metrics. In order to be complete, to explain all issues concerning Stats, Stats and the Index Metrics touched by those commands are documented here. The Index Metrics are not covered in detail here because they are metrics, not Stats, and they belong with the index documentation. All issues concerning indices are fully explained in Software Gems' *Sybase Data Storage & Fragmentation* document (PDF or HTML), and are reported in our *HelpIndex* & *HelpIPartition* utilities.

## 2.3 Histogram

The purpose of an Histogram is to describe the distribution of data values in a **single column** of a table. An Histogram is made up of **Cells**. These Cells are generated automatically when certain UPDATE STATS commands are executed, based on the operands of the command.

### 2.3.1 Histogram Type

The first issue to be understood is, Histograms have limitations 4:

- The type of Histogram used in ASE is 'equi-height' (equi-depth, equi-sum), which means:
  - each Cell is expected to describe roughly the same proportion of the table,
  - the proportion (percentage of the population represented) of each Cell must be evenly distributed, otherwise the Histogram as a whole (as opposed to the content of particular Cells) is skewed

### 2.3.2 Cell Content

Each Cell represents an [equal] portion of the [Values in a single column of the rows in] a table. Thus the purpose of each Cell is to store:

data distribution Stats, for that portion of the [Values in a single column of the rows in] a table.

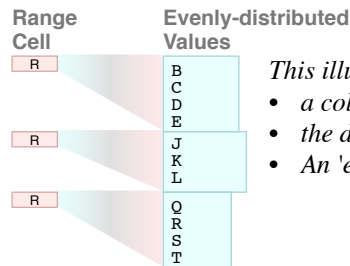
There are two types of Cells (not three or four, as commonly discussed). This may not be easy to differentiate from the optdiag report, let alone the hilarious manual or the confused writings on the web.

### 2.3.3 Range Cell

There are three renditions of the Range Cell: the normal; and two special cases (they are not discrete types of Cell).

#### Range Cell

- In optdiag this is depicted with "<=".
- Identifies the **range of Values** for that Cell (portion of the table), hence the name.
- The upper bound of the range is the Value of the Cell; the lower bound is the Value (upper bound) of the previous Range Cell (ie. the Values are not duplicated).
- The **Weight** is the percentage of the rows that fall within the range of Values.



This illustrates:

- a column with values that are evenly distributed (not perfect!)
- the distribution of such values is dense 6
- An 'equi-height' Histogram services the 'equi-height' range of values

#### Range Cell/Null

- The first Cell in the Histogram is a Range Cell for **Null Values**, it identifies the percentage of Null Values in the column. It is always present, regardless of whether the column allows Nulls.
- The Value *shown* is the lowest Value in the column, because it is a Range Cell, and that is the Value that Null is "<=" less-than-or-equal-to (do not let *what Null "represents"* confuse you, the first Cell represents Null Values).

#### Range Cell/Placeholder

- The second type of Cell, a Frequency Cell (wait for it), requires an empty Range Cell in order to establish its existence (detailed in the Catalogue section).
- In optdiag they are depicted with "<".
- Such Range Cells have a zero Weight and a Null Value

The 'equi-height' Histogram, with its Range Cells, works well, they are quite adequate for data values that are:

- distinct 5, or close to distinct, regardless of distribution
- highly repeated 5 Values that are evenly distributed (ie. not badly skewed)
- Values that are Dense 6, whether evenly distributed or not, whether repeated or not.

But it is inadequate when the Values are non-distinct (repeated) *and* unevenly distributed (badly skewed) *and* sparse.

Eg. imagine a column that contains 50 StateCodes:

- where the data is evenly populated and distributed, the average percentage of the column, regardless of table population, for each StateCode is 2%.
- where the population of some StateCodes is substantially different from the average 2%, the data is uneven.
- where a few StateCodes populate far more than 2% of the column, the data is badly skewed.

That is, Range Cells are great for all data except highly-repeated Values which are sparse, which is not an uncommon occurrence (eg. a Foreign Key to a Reference or Lookup table). For this, we need a Frequency Cell.

4 If the table is in fact a record file in a Record Filing System, non-relational, subject to substantial change in population, there is no point in agonising over accuracy or freshness of the Stats: each execution of UPDATE STATS will create an entirely new set of Stats due to the entirely new data population. (Relational Database tables are stable, and ASE is a Relational Database platform.)

5 Terms such as 'unique' and 'duplicated' are incorrect and confusing, because they apply to DataStructures, and are established terms, with strong connotations: Such non-technical labelling leads to idiocies such as a *DataStructure that is both 'unique' and 'duplicated' at the same time*. Here we are describing distribution, the set of Values in a data column *per row*, which has little to do with whether the DataStructure is 'unique' or not:
 

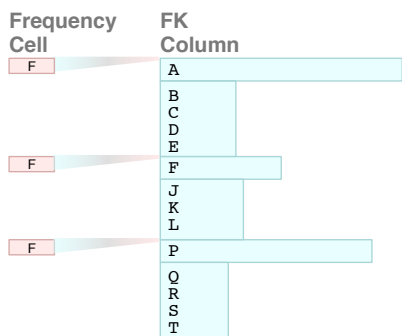
- while 'unique' may be set at the DataStructure level, a column Value may well be repeated per row
- 'unique' applies to a set of columns that make up a key, not to a single column (unless of course the key is a single column)
- 'duplicated' is incorrect, false, the Value is merely repeated per row.
- DISTINCT is precisely understood by those who know SQL.

6 **Dense** and **Sparse** are technical terms, not limited to Histograms.
 

- Sparse means there is a substantial gap between the Values, eg. {A; F; P; X }.
- Dense means there is little or no gap between the Values, eg. consecutive numbers.
- Dense and Sparse apply to the distribution of data Values: it is neither a type of cell, nor a type of frequency.

### 2.3.4 Frequency Cell

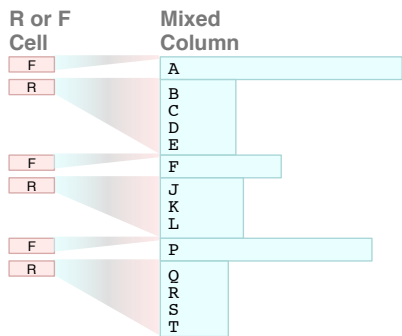
- In `optdiag` this is depicted with "="
- The purpose of Frequency Cells is to expose such highly-repeated Values, where the Value diverges substantially from the distribution profile.
- This is a method of providing for highly-repeated Values, within a structure that was designed for proportionate Ranges: Frequency Cells (ie. from an 'equi-width' Histogram), are an enhancement to the otherwise ordinary 'equi-height' Histogram. The result is, the ASE Histogram allows mixed Cells: 'equi-height' for proportional distribution and 'equi-width' for sparse, repeated-value distribution.
- The **Value** is the *single* highly-repeated Value (whereas the Value in a Range Cell is a range boundary)
- The **Weight** is the percentage of the rows that have that single Value
- The Frequency Cell is preceded by a Placeholder Range Cell <sup>7</sup>



This illustrates:

- a column that is mostly evenly distributed
  - the distribution of such values is dense <sup>5</sup>
- with two highly-repeated values A and P
  - the distribution of such values is sparse <sup>5</sup>
- and one skew value F
  - Frequency Cells that are required to service such values

Thus the Sybase ASE Histogram can certainly cater for a variety of distribution types in a column: evenly-distributed values; various types of unevenly-distributed values; and repeating values, including skew values. In other words, it can handle both dense and sparse distribution in the same column.



This illustrates:

- Range Cells that cover the evenly and unevenly distributed values, and
- Frequency Cells to serve the highly-repeated and skew values

The administrative task demanded is to ensure that the Histogram has enough Cells, and of the right type, based on the actual content of the column. That it covers:

- Dense distribution
  - it has enough Range Cells to cover the overall range of values, and to cover it adequately, and
- Sparse distribution
  - it has Frequency Cells that cover the skewed values.

Otherwise the [default] Stats would be quite inadequate, and cause the Optimiser to make decisions that are *not* based on the actual content of the column.

#### The Name

The name Frequency Cell *grates*. It is properly a Distribution Cell. It identifies the Weight of a single Value, whereas the Range Cell identifies the Weight of a range of Values. Since the latter is named a Range [of Values] Cell, and it is correct, the former should be named a Single Value Cell.

### 2.3.5 Partitioned Table

The documentation is especially hopeless in this area. For Partitioned tables, for which an Histogram is generated on a column (via any flavour of `UPDATE STATS` which implies such), the Histograms generated are (a) one for the table level, and (b) one for each Partition. Note that the Steps are specified either at the table level, applying to both table and partitions, or at partition level, applying to the given partition only: eg. for a large table with 16 Partitions, if 200 Steps is specified, and no partition is specified, the number of Cells generated is 17 x 200. Therefore for Partitioned tables, it is particularly important to ensure that the number of Cells is not some large unexpected number.

### 2.3.6 Automatic Tuning

One consequence of this messy state of affairs, due to having to use a high number of Steps, in order to get ASE to recognise skew Values, and build Frequency Cells (Single Value Cells) for them, is that the number of real (not placeholder) Range Cells generated may be too many (ie. their range of Values or Weights may be too small to be relevant). Thus a number of consecutive Range Cells can be collapsed into a single one. This is performed automatically by ASE, depending on histogram tuning factor, there is nothing to do <sup>8</sup>.

<sup>7</sup> In the first few releases, we had only Range Cells. The engineer who added Frequency Cells evidently wanted to leave us something to remember him for.

<sup>8</sup> The setting of 1 means disabled; the default is 20. "Do not change it unless asked to do so by Technical Support."



### 2.4 ColumnGroup Stats

This section discusses Stats for groups of columns taken together. There are two types:

- ColumnGroups/Indexed: these are established implicitly via Index Stats
- ColumnGroups/Not Indexed: established explicitly, via ColumnGroup Stats

#### 2.4.1 ColumnGroup/Indexed

For Each	Sybase Can Store	Also known as
ColumnGroup	ColumnGroup Stats	ColumnGroup Density
Column	Column Stats	Column Density
	Histogram	Distribution

Stats can be rendered for Indices. Where an Index consists of many columns, as is normal for Relational databases <sup>4 9</sup>, Stats can be rendered for:

- the set of columns, taken together; and
- for each increment in the set

Where an Index such as:

```
U_ComponentMetric ( Server, Component, Metric, DateTime )
```

is implemented, the columns and sequence are chosen purposely, on the basis that queries using:

- Server
- Server, Component
- Server, Component, Metric
- Server, Component, Metric, DateTime

are expected. These are column **increments** within the ColumnGroup. Of the four increments, the first is of course a single column (Column Stats and an Histogram are possible), and the next three are **ColumnGroups** themselves (ColumnGroup Stats are possible). There are no Histograms per ColumnGroup, because they are more relevant at the column, not ColumnGroup, level; ColumnGroup Densities are deemed adequate; and additional Histograms would require even more time to build, and process.

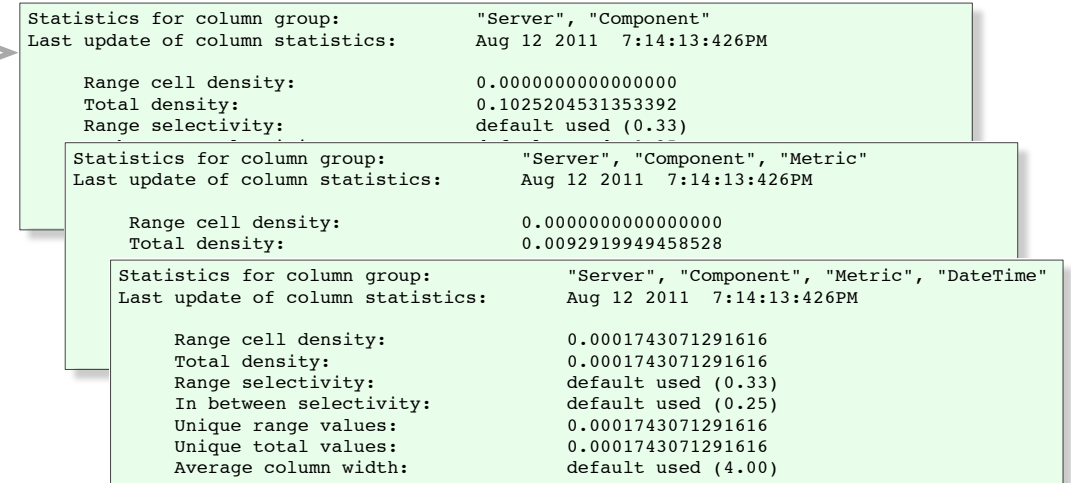
#### 2.4.2 ColumnGroup/Indexed Stats

Stats for Indexed ColumnGroups are generated, via (full detail at [4.4]):

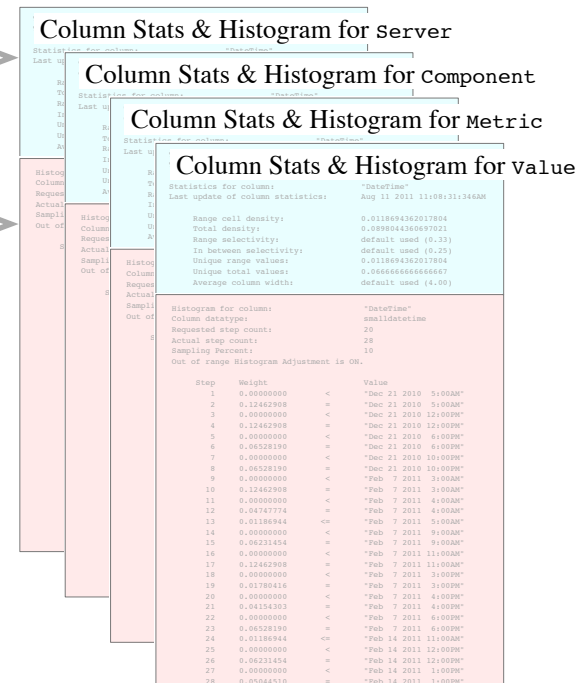
```
UPDATE INDEX STATS Table [Index]
```

For each second and subsequent increment in an Indexed ColumnGroup (the first increment is covered as a single column [2.2]), ASE generates:

1 Density and Selectivity Stats, as reported in optdiag:



2 Column Stats and an Histogram for all columns in the Index <sup>10</sup>.



<sup>9</sup> Note that if you have only surrogates (Record IDs) for "keys", they are not Relational Keys, and the "database" is in fact not a database but a 1960's Record Filing System, deployed in a database container for convenience (SQL access; recovery; etc). Such systems do not have any of the integrity, power or speed of a Relational database. There is no point wasting time trying to "improve" Stats on such RFS: it is like installing a turbo-charged gasoline engine in an ox cart, you will spend half your life trying to "improve" the speed while ensuring the wheels don't fall off. Upgrade to a car (any car) and you won't have to limit the speed.

<sup>10</sup> If UPDATE STATS Table Index (older form, now deprecated for Indices) is used, ASE generates Density and Selectivity Stats for each second and subsequent increment in the Indexed ColumnGroup: and Column Stats and an Histogram for the leading column only.

### 2.4.3 ColumnGroup/Not Indexed

For Each	Sybase Can Store	Also known as
ColumnGroup	ColumnGroup Stats	ColumnGroup Density
Column	Column Stats	Column Density
	Histogram	Distribution

As ASE progressed, two things were realised:

- that Stats should not be tightly related to Indices (which may be dropped and recreated, or replaced, whereas columns are permanent, and their Stats prevail, thus they should remain, even if the related Index is not recreated)
- that there are ColumnGroups that are frequently in queries, that are *not* Indexed, and Stats are required for them.

Therefore, for decades, Indices have not been tightly related to Stats, and ColumnGroups are supported either with an index that identifies them, or without. There *is* a relation between columns (for which Stats are collected, and an Histogram is constructed) and Indices, but it is fluid, not a direct reference via `indid`.

It is important to understand and use ColumnGroups that are not part of ColumnGroup/Indexed increments; and not members of an Index, and to establish Stats for them. Any group of columns that are frequently used in queries can be declared to ASE as a ColumnGroup, such that Stats are captured for them.

### 2.4.4 ColumnGroup/Not Indexed Stats

ColumnGroups are explicitly declared, and Stats are generated, via (full detail at [4.3]):

```
UPDATE STATS Table (Column_1, Column_2, ...)
```

For each second and subsequent increment in an Indexed ColumnGroup (the first increment is covered as a single column [2.2]), ASE generates:

1 Density and Selectivity Stats, as reported in `optdiag`:

```
Statistics for column group:      "Value", "Metric"
Last update of column statistics: Aug 12 2011  8:51:12:780PM

Range cell density:              0.0032336605036902
Total density:                  0.0037069975081228
Range selectivity:              default used (0.33)
In between selectivity:         default used (0.25)
Unique range values:            0.0030454474465094
Unique total values:            0.0032051282051282
Average column width:           default used (20.00)
```

2 Column Stats and an Histogram for the leading column only.

```
Column Stats & Histogram for value
Statistics for column:           "DateTime"
Last update of column statistics: Aug 11 2011 11:08:31:346AM

Range cell density:             0.0118081262017804
Total density:                  0.0588044309097021
Range selectivity:              default used (0.33)
In between selectivity:         default used (0.25)
Unique range values:            0.0118081262017804
Unique total values:            0.0666666666666667
Average column width:           default used (4.00)

Histogram for column:           "DateTime"
Column datatype:               smalldatetime
Requested step count:           10
Actual step count:              28
Sampling Percent:               10
Out of range Histogram Adjustment is ON.

Step  Weight  Value
-----  -
1  0.00000000  < "Dec 21 2010 5:00AM"
2  0.12462908  = "Dec 21 2010 5:00AM"
3  0.00000000  < "Dec 21 2010 12:00PM"
4  0.12462908  = "Dec 21 2010 12:00PM"
5  0.00000000  < "Dec 21 2010 6:00PM"
6  0.06528190  = "Dec 21 2010 6:00PM"
7  0.00000000  < "Dec 21 2010 10:00PM"
8  0.06528190  = "Dec 21 2010 10:00PM"
9  0.00000000  < "Feb 7 2011 3:00AM"
10 0.12462908  = "Feb 7 2011 3:00AM"
11 0.00000000  < "Feb 7 2011 6:00AM"
12 0.04747774  = "Feb 7 2011 6:00AM"
13 0.01180844  = "Feb 7 2011 9:00AM"
14 0.00000000  < "Feb 7 2011 9:00AM"
15 0.06231454  = "Feb 7 2011 9:00AM"
16 0.00000000  < "Feb 7 2011 11:00AM"
17 0.12462908  = "Feb 7 2011 11:00AM"
18 0.00000000  < "Feb 7 2011 3:00PM"
19 0.01780416  = "Feb 7 2011 3:00PM"
20 0.00000000  < "Feb 7 2011 6:00PM"
21 0.04154303  = "Feb 7 2011 6:00PM"
22 0.00000000  < "Feb 7 2011 6:00PM"
23 0.06528190  = "Feb 7 2011 6:00PM"
24 0.01180844  = "Feb 14 2011 11:00AM"
25 0.00000000  < "Feb 14 2011 12:00PM"
26 0.04231454  = "Feb 14 2011 12:00PM"
27 0.00000000  < "Feb 14 2011 1:00PM"
28 0.05044510  = "Feb 14 2011 1:00PM"
```

In this example, the command used was:

```
UPDATE STATS SM_ComponentValue ( Value, Metric )
```

for which ASE generated ColumnGroup Stats (above), and Column Stats and an Histogram for the `Value` column only [2.2]. If Histograms are desired for the second and subsequent columns in the ColumnGroup *and* they have not been generated as a result of their participation in some Index [2.4.1], use:

```
UPDATE STATS Table ( ColumnName )
```

- Actually, there are a few more Column and ColumnGroup Stats, but we will leave that for now

# Sybase Statistics Demystified

## 2.5 Concept • optdiag Stats



This section discusses the ASE reports that can be obtained for determining and analysing Stats.

For Each	Sybase Can Store	Also known as
ColumnGroup	ColumnGroup Stats	ColumnGroup Density
Column	Column Stats	Column Density
	Histogram	Distribution

```

Statistics for column group:      "Server", "Component", "Metric", "DateTime"
Last update of column statistics: Aug 12 2011  7:14:13:426PM

Range cell density:              0.0001743071291616
Total density:                  0.0001743071291616
Range selectivity:              default used (0.33)
In between selectivity:         default used (0.25)
Unique range values:            0.0001743071291616
Unique total values:            0.0001743071291616
Average column width:           default used (4.00)
    
```

The Sybase utility `optdiag` reports Stats that have been generated and stored, as follows. (It may be compared with our utilities, expanded in chapter [6]):

- Density and Selectivity Stats on ColumnGroups (right)
- Density and Selectivity Stats on Columns (below)
- Distribution Stats for Columns in the form of an Histogram

```

Table owner:      "dbo"
Table name:      "SM_ComponentValue"

Statistics for column:      "DateTime"
Last update of column statistics: Aug 11 2011 11:08:31:346AM

Range cell density:      0.0118694362017804
Total density:          0.0898044360697021
Range selectivity:      default used (0.33)
In between selectivity: default used (0.25)
Unique range values:    0.0118694362017804
Unique total values:    0.06666666666666667
Average column width:   default used (4.00)
    
```

```

Histogram for column:      "DateTime"
Column datatype:         smalldatetime
Requested step count:     20
Actual step count:       28
Sampling Percent:        10
Out of range Histogram Adjustment is ON.
    
```

Step	Weight	Value
1	0.00000000	< "Dec 21 2010 5:00AM"
2	0.12462908	= "Dec 21 2010 5:00AM"
3	0.00000000	< "Dec 21 2010 12:00PM"
4	0.12462908	= "Dec 21 2010 12:00PM"
5	0.00000000	< "Dec 21 2010 6:00PM"
6	0.06528190	= "Dec 21 2010 6:00PM"
7	0.00000000	< "Dec 21 2010 10:00PM"
8	0.06528190	= "Dec 21 2010 10:00PM"
9	0.00000000	< "Feb 7 2011 3:00AM"
10	0.12462908	= "Feb 7 2011 3:00AM"
11	0.00000000	< "Feb 7 2011 4:00AM"
12	0.04747774	= "Feb 7 2011 4:00AM"
13	0.01186944	<= "Feb 7 2011 5:00AM"
14	0.00000000	< "Feb 7 2011 9:00AM"
15	0.06231454	= "Feb 7 2011 9:00AM"
16	0.00000000	< "Feb 7 2011 11:00AM"
17	0.12462908	= "Feb 7 2011 11:00AM"
18	0.00000000	< "Feb 7 2011 3:00PM"
19	0.01780416	= "Feb 7 2011 3:00PM"
20	0.00000000	< "Feb 7 2011 4:00PM"
21	0.04154303	= "Feb 7 2011 4:00PM"
22	0.00000000	< "Feb 7 2011 6:00PM"
23	0.06528190	= "Feb 7 2011 6:00PM"
24	0.01186944	<= "Feb 14 2011 11:00AM"
25	0.00000000	< "Feb 14 2011 12:00PM"
26	0.06231454	= "Feb 14 2011 12:00PM"
27	0.00000000	< "Feb 14 2011 1:00PM"
28	0.05044510	= "Feb 14 2011 1:00PM"

- The Histogram stores detailed information re the actual data content (**Values**) of a column, and its **distribution**, for the whole table. Histograms are not generated for ColumnGroups.
- The **Number of Values** is the number of parts that the table is divided by, for statistical purposes, as initially *requested* in: `UPDATE STATS USING n VALUES`
- The default for which is 20, and the default itself can be set via `number of histogram steps`.
- Once `n VALUES` has been set for a table, it remains.
- The components of the Histogram are called **Cells**, they are generated from the actual data, and they are stored. The requested `n VALUES` identify an intention, an upper bound; the number of Cells generated is based on the actual values in the column.
- Note that the item reported as "Step" in `optdiag` is obviously incorrect, it is a Cell.

**Cell vs Step vs Values**  
 If `n VALUES` is understood, the entire notion of "Steps"; the multiple definitions; the consequent labouring over what it is; what the differences are, can be dismissed.

**The Name**  
 Those who are new to Sybase ASE often wonder about the name. Of course, it is historic. In the early days, the central focus of query performance was the Optimiser, the choices it made, which was based on (a) Index Metrics and (b) Statistics. The utility was intended to diagnose the Optimiser, hence the name. It also explains why (a) and (b) were reported together. The Optimiser has advanced substantially, it is no longer based on Statistics alone, its behaviour depends on a variety of resources, which are each constrained by configuration parameters. Diagnosis is no longer limited to Statistics: each resource type has a method for diagnosis.





2.6.1 Summary

First, a summary of the default operation (20 VALUES):

- For any given range-of-values in a column, a number of Range Cells are generated, each defining a Range of Values and the percentage of qualifying rows, plus one for Nulls. That number is:
  - a. initially identified by the requested VALUES, an intention, and
  - b. by the actual Cells generated, which may be less.
- For Partitioned tables, where the partition is not specified, that number is:
  - c. multiplied by the number of Partitions plus one (for the table level).
- Additionally, if the data in the column is badly skewed, for those highly-repeated, sparse Values that ASE identifies, within the given number of VALUES, Frequency Cells (Single Value Cells) are generated, with placeholder Cells.

2.6.2 Cell Requirement

We are now ready to determine the two important metrics in Statistics: the Cells required; and the type of Cells. That requires knowledge of the actual Values in the table, on a column basis of course, and how they are distributed. There is no substitute for *Know Your Data*, it is essential to *all* aspects of database design and administration, no less *the* aspect of Statistics, where it is central. Execute this code snippet for each column that is deemed to require Statistics:

```
SELECT DISTINCT
    Column,
    COUNT( Column )
FROM Table
GROUP BY Column
```

The determination of the optimal number of Cells (VALUES), is neither an arduous task, nor a black art: it is straight-forward science, and only requires a bit of testing and verification on your part. The default 20 VALUES is certainly inadequate for tables of any reasonable size, and it must be increased to suit the actual distribution. However, do not tend towards the maximal, because the Cells are stored in *sysstatistics*, loaded into ASE memory structures (once), and inspected by the Optimiser every time a query plan is compiled.

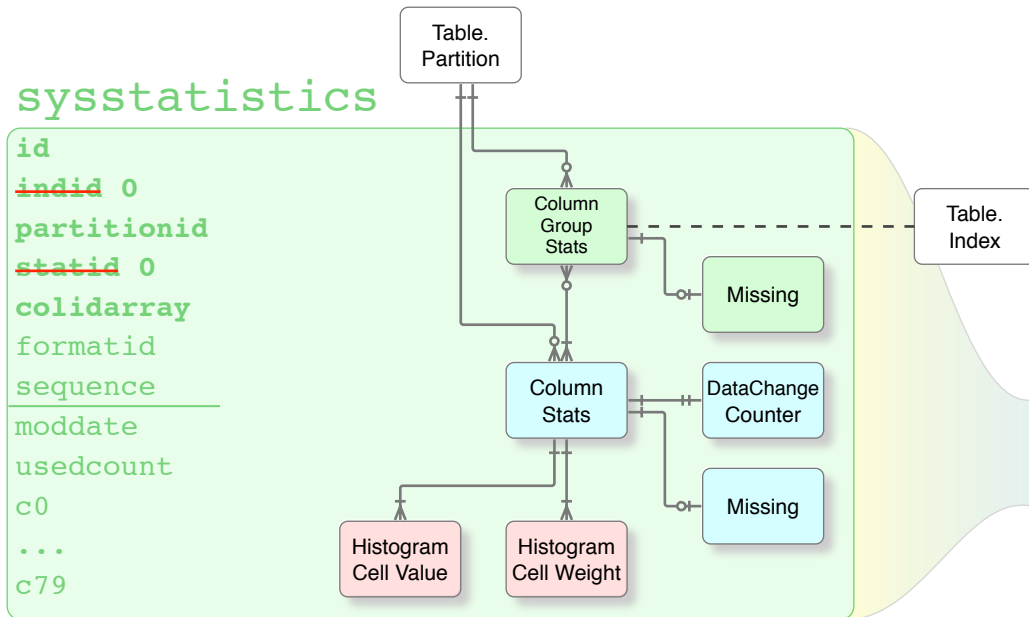
This provides a guide for the optimal number of Cells.

Column Value	Sparse/ Dense	Cell Type Required	Num Cells	
Distinct	Dense	Range	Based on <Rows>	<i>To obtain 'enough' Cells</i>
	Sparse			
Non-distinct (Highly repeated)	Dense	Range	<DistinctValues> + 1	<i>If Range Cells deemed adequate</i>
	Sparse	Frequency	<DistinctValues> x 2	
		Range	<DistinctValues> ÷ 2	
Each Skew Value	Sparse	Frequency	<SkewValues> x 2	

### 3.1 The Logical

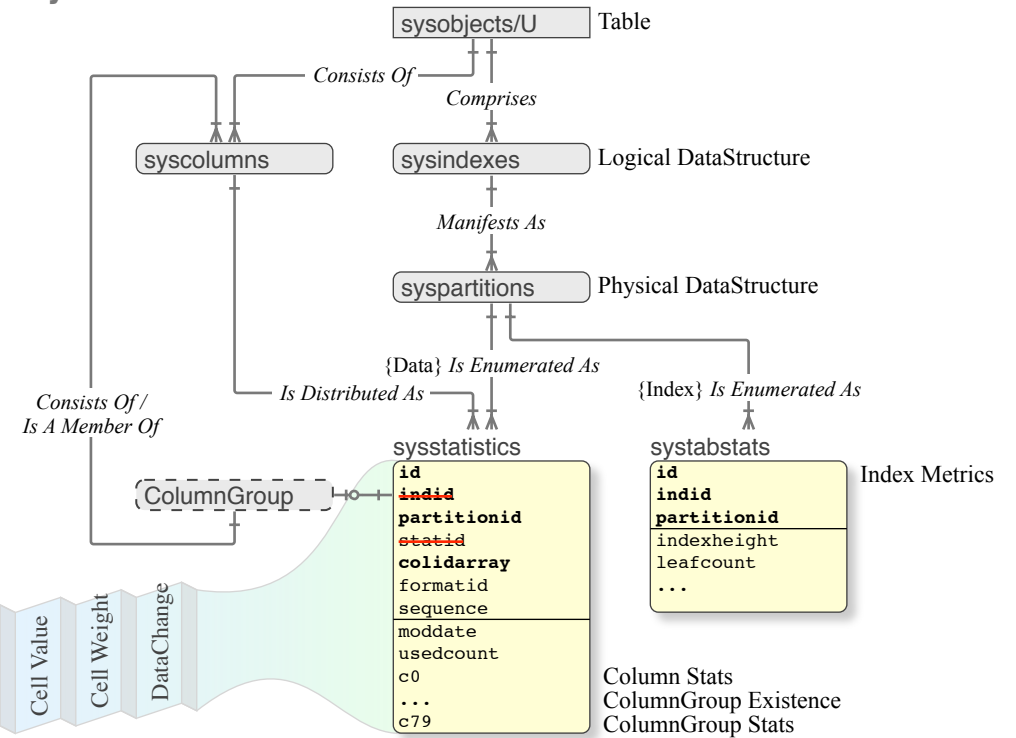
Performing anything more than perfunctory administration requires some understanding of the *Sybase* Catalogue. It is invaluable in understanding the server, and appreciating why tasks must be performed in certain ways, why they cannot be performed as the newcomer thinks they "should" be. Statistics is no exception. Demystifying Statistics is roughly equivalent to demystifying the Statistics catalogue.

The easiest, and the best method of understanding data, and how the data relates to each other, is to model it according to the *Relational Model*<sup>11</sup>. Nine types of data, or distinct *logical* tables, seven of which are expanded, are stored in `sysstatistics`. Their relations, at Entity-Relation level, are:



- ASE maintains Stats at both the table level (`partitionid = 0`) and partition level.
- **Columns** must exist for a table, they are the central focus of Stats
- **ColumnGroups**, however, may not exist, since there are people in the new *Sybase* community who do not have Relational databases; nor implement Stats for Indices, nor for columns that are queried together. If they do exist, they are made up of columns, and a column may exist in more than one ColumnGroup.
- **Histogram Cells** are stored per column [that has Column Stats]. This is the bulk in the table.
  - Histogram Cell **Value & Weight** are stored in separate vectors, with up to 80 values per row.
- **Missing** per Column or ColumnGroup Stats constitute logging of events.
- **DataChange Counters** per column are just that, not really Stats, in a convenient storage location.
- Like ColumnGroups, Indices may not exist, but if they do, they are made up of columns, and a column may exist in more than one **Index**. If Stats are captured for them, it will be in the form of a ColumnGroup. That is the fluid relation between ColumnGroup Stats and Index, the Index Key is the ColumnGroup.
- The nine types of data, or logical tables are differentiated by `formatid` (expanded on the next page).

### 3.2 The Physical



Describing this internal table as "denormalised" is incorrect: the *data* is beautifully Normalised [3.1]. However, nine logical tables (*types* of rows) are stored in the one physical table. It is therefore nine logical tables *folded* into one physical table, and a component in the Key differentiates them.

- That is unacceptable in a Relational database, but in an internal catalogue of a server, particularly one that contains a large amount of varied data (note the actual data Values in the Cells, and the volume of Cells), it is quite desirable.
- The strictly Normalised version would be nine tables, *plus* one table for each Datatype.
- Therefore, implementing `colidarray` as a *variable length* column to contain any Datatype is warranted. However, doing so *in an index*; and storing Null Values, is a crime against nature which is hard to forgive.
- Notice there is no direct relation to `sysindexes`, despite the presence of the `indid` column.

<sup>11</sup> Familiarity with the *Sybase ASE* catalogue will be helpful. Familiarity with our *Sybase ASE Data Model* (which contains important explanations and additional facilities) will assist even further.

### 3.3 Row

This section defines the rows, the nine physical tables (seven that are expanded) in `sysstatistics`.

#### 3.3.1 Key

Every row in a Relational Database must be unique, and identified by its Primary Key. Accordingly, the Primary Key columns identify the `sysstatistics` row content:

Column	Content	
<code>id</code>	Identifies the Table, Foreign Key referring to <code>sysobjects</code>	
<code>indid</code>	Zero. Identifies the data portion of a table (leaf level of a Clustered Index, or a Heap)	
<code>partitionid</code>	zero	Table level Stats
	non-zero	Partition level Stats; Foreign Key referring to <code>syspartitions</code>
<code>statid</code>	Not used	
<code>colidarray</code>	1 entry	Data applies to the single identified column, Foreign Key ( <code>id</code> , <code>colid</code> ) refers to <code>syscolumns</code>
	>1 entry	List of ordered colids, identifying a ColumnGroup. Data applies to the ColumnGroup. One Foreign Key per entry ( <code>colid</code> ), referring to <code>syscolumns</code>
<code>formatid</code>	Type of Stats row (together with <code>colidarray</code> ), as per the table below.	
<code>sequence</code>	Where the data items exceeds 80, sequence identifies rows in the series.	

- Ensure your code is immune to possible changes in the future:

```
SELECT ...
FROM sysstatistics
WHERE indid = 0
AND statid = 0
```

#### 3.3.2 Row Type

The nine logical tables (*types* of rows) in `sysstatistics` are identified by (`formatid`, `colidarray`), their content is as follows. Additional Stats that are constructed for Simulations are not expanded in this document:

<code>formatid</code>	<code>colidarray</code>	Pertains to	<code>c0...c79</code>	
20			Simulated Stats	<i>Not expanded</i>
40			Simulated Server config parm	<i>Not expanded</i>
100	1 entry	Column	Column Stats	
	>1 entry	ColumnGroup	ColumnGroup Stats	
102	1 entry	Column	Up to 80 Histogram Values	<i>Refer sequence for more</i>
104	1 entry	Column	80 Histogram Weights	<i>Refer sequence for more</i>
108	1 entry	Column	DataChange Counter	
110	1 entry	Column Stats Missing		
	>1 entry	ColumnGroup Stats Missing		

The usage and content of the attributes are dependent on the row type (`formatid`, etc):

<code>moddate</code>	Timestamp of the last update of this row
<code>usedcount</code>	The number of data items [ <code>c0...c79</code> ] used in this row
<code>c0...c79</code>	Up to 80 data items, content as above

- Histogram Cell Values (variable, data dependent) and Histogram Cell Weights (fixed, small) are stored on separate vectors, per `formatid`: the sequence and rows of Cell Values do not match that of Cell Weights. Nevertheless, the *aggregate* `usedcounts` are the same.

#### Warning

- *I have seen at least four different methods of 'interpreting' `colidarray`: every one of them uses procedural processing, which, for a set-processing engine, is (a) very slow, and (b) quite unnecessary. Additionally, they are all wrong, in that every one of them takes a laborious approach to unpacking `colidarray`, and fails miserably. The task is simple, if one understands that it is simply an array of words (two bytes), each word being a single `colid`.*
- *One precious darling handles endian issues **manually**; in Sybase ASE that is not only unnecessary, it is stupefying.*

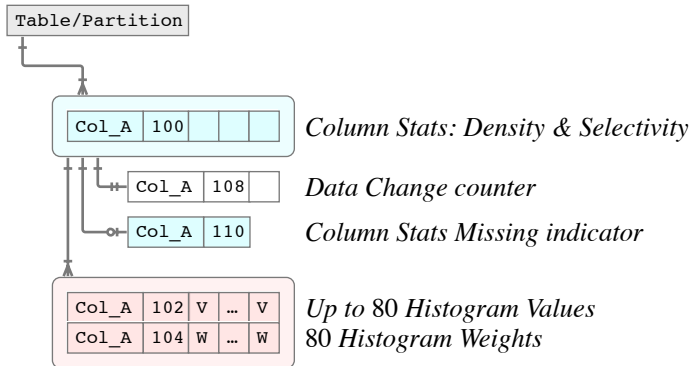
As with all our software, both ***HelpStatistic*** and ***HelpHistogram*** use the set-processing power of the Relational engine, and thus execute at set-processing speeds, not at developer speeds. The absence of the various failures found in free "software" (crashes as well as false reports), and the speed (which is several orders of magnitude slower), differentiate ours from such.

### 3.4 Statistics

This section defines the content of `sysstatistics` rows that contain Stats.

#### 3.4.1 Column Stats

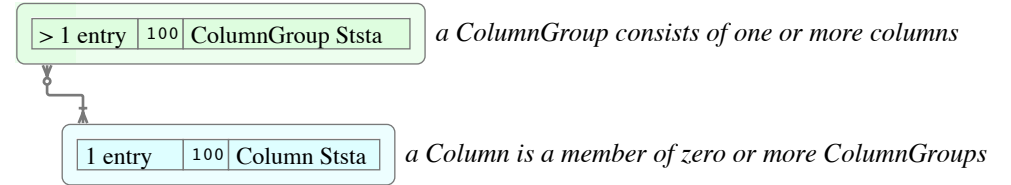
It may be easier to understand in a relational diagram showing example rows. Let's look at Stats for a single column first: `colidarray` consists of a single entry or an array of [1], a single `colid`, the PK of its parent, `syscolumns`.



- The Cell Value and Cell Weight together comprise a single Cell in the Histogram.

#### 3.4.2 ColumnGroup Stats

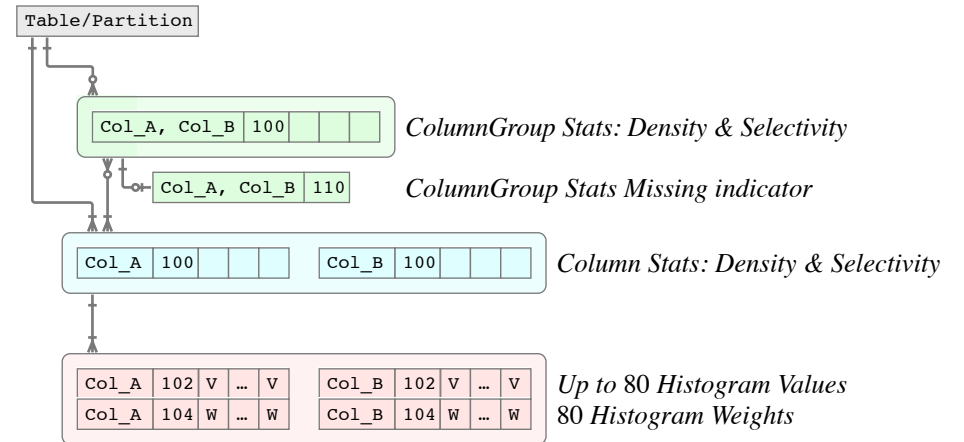
Now let's tackle the ColumnGroup. Ensure that you understand [2.4] **ColumnGroup Stats**. Now we are concerned with the Primary Key: a single `colidarray` entry identifies Column Stats; more than one `colidarray` entries identify ColumnGroup Stats.



Just as the `colid` identifies a column (within a table), the `colidarray` identifies an *array* of `colids` (within a table). The original Engineers have used the Relational concept of Identifiers brilliantly (rare nowadays): the Identifier of a ColumnGroup is the list of Identifiers of its component columns. They reference each other without additional Identifiers, and eliminate the need for an Associative table (or, the Foreign Keys do exist, and it is folded into the Primary Key, in `sysstatistics`, without repetition). (Again, that is acceptable in a server catalogue, but not in a database.)

With a relational diagram showing example rows, we can immediately identify:

- a ColumnGroup row (`colidarray` > 1 entry)
- the members of ColumnGroup (contained in the ColumnGroup Identifier itself)
- the member Column rows (`colidarray` of 1 entry)



- There is no Histogram for ColumnGroups, the Histogram for each column is used.

### 3.5 Histogram

This section defines the content of `sysstatistics` rows that contain Histograms.

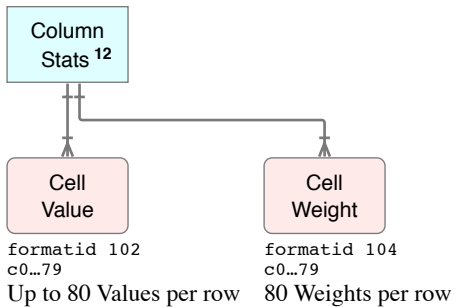
Since actual data Values are stored in the Cells, the varying Datatype and the length are an issue, thus the storage is minimised and optimised. Histogram Cells take up the bulk of storage in `sysstatistics`. When a query is compiled and Optimised, it is read in the Procedure Cache and used, along with Sort Buffers. Therefore it is the single most important set of metrics re Stats (if one had to pick one) that the administrator should address: that the number of Cells is optimal (neither too few nor too many); and that only columns that are actually used *frequently* in queries have Histograms. And that is correct for Partitioned tables in particular.

When Histograms were implemented in *Sybase ASE 12* there were only Range Cells. Frequency Cells followed later. The problem is, there is no way to differentiate them from Range Cells, or to identify them as Frequency Cells (a `CellType` column has not yet been added to the catalogue). The method used to implement Frequency Cells without changing the extant structure is, a Range Cell with no Weight and a Null Value is used to identify that the *next* Cell is a Frequency Cell. Thus the Frequency Cell requires a *placeholder* Range Cell.

That is the reason for the very silly `optdiag` report, in which every Frequency Cell is preceded by a zero-weight Range Cell [2.3.3]. For the placeholder Range Cell, no Value stored (a Null Value means no Value), which is desirable, since the Value is stored as `varbinary`, and that space can be eliminated. But for insanity reasons (probably to elevate the stupidity of printing of empty Range Cells), `optdiag` replicates the *Frequency Cell Value* in the *placeholder* Range Cell, which in fact does not have a Value.

Those who inspect the Cells directly should note a second complication. Due to the tight packing of Cell data into `sysstatistics`, in order to minimise the space used for Cells, and to allow Weights and Values to be updated independently, the structure used for storage does not reasonably match the logical structure (in terms of understanding what Cells are, and identifying the Cell type accessed). Much of the structural form exists (a) for historic reasons, and (b) to support the normal case, rather than the exceptional case.

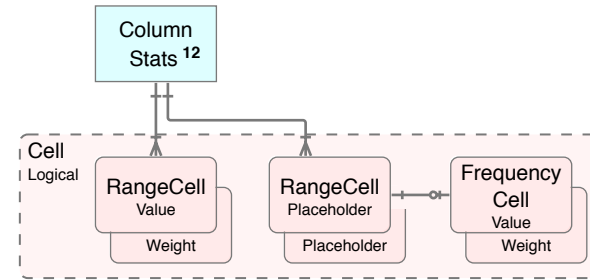
#### Current Storage (Physical)



This is the actual storage for the Histogram, with separate Value and Weight vectors, due to their different widths, which must be navigated independently. The Cell type is not known beforehand. Due to Values being large and Weights being tiny, the row counts for the two vectors are quite different, but the cumulative `usedcount` (per row) is the same. There is actually one overall Cell count stored elsewhere.

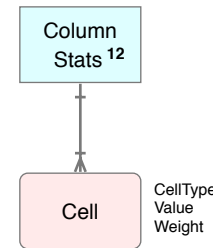
However, that does not show the logical structure, which needs to be understood for accessing Cells directly and determining the Cell type: a Frequency Cell (Single Value Cell) and a Range Cell are mutually exclusive, but a Frequency Cell depends on a placeholder Range Cell to establish its existence.

#### Actual Storage (Logical)



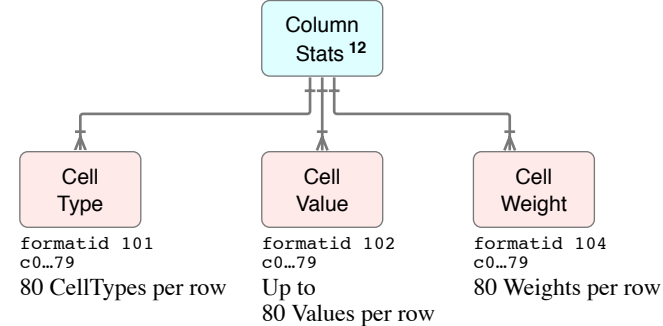
Reading in a large number of empty Cells (placeholders) is not funny, even if a legacy, and begs address. The rather obvious next enhancement is to Normalise the storage by adding a `CellType` discriminator. Of course the storage would be substantially more efficient, and the Optimiser (all queries) as well as `UPDATE STATS` would be faster.

#### Some Future Version (Logical)



Due to the Values being varying; wide; and uncontrollable, versus the small and controllable Weights, the engineers would require separate vectors for Value vs Weight, as per the current version.

#### Some Future Version (Physical)



<sup>12</sup> A column that has Column Stats, and therefore an Histogram, ie. the Cell is not optional.



# 4.1 Example

## 4.1.1 Table

This chapter details all considerations for each flavour of the UPDATE STATISTICS command. A moderately complex example is useful to assist understanding, it is detailed here.

### Table & Indices

SM\_ComponentValue

Server
DateTime
Component
Metric
Value

```
UC_PK (
  Server,
  DateTime,
  Component,
  Metric
)
```

```
U_ComponentMetric (
  Server,
  Component,
  Metric,
  DateTime
)
```

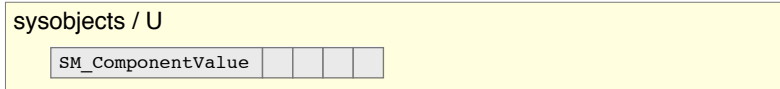
- A simple, Data Warehouse table (columnar Normalisation; Fact-Dimension; full pivot), with two indices
- The table is APL, therefore the Clustered Index is real, and Range Queries are supported **49**
- The equivalent DOL file would be too simple (in its base state) to qualify as an example, and if "improved" to provide a fraction of Relational capability, it would be complex and repetitive, due to such having far more indices than the equivalent Relational table
- U means unique; C means clustered.

## 4.1.2 Catalogue

The rows in the Sybase ASE catalogue for this table, that are relevant to Statistics operations, are as follows. The relations between Columns and ColumnGroups, and to indices and index metrics are also shown:

### Table Definition

Existence



### DataStructure · Logical

Structure; Components various



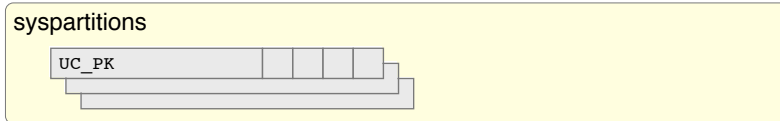
### Index Definition

Structure; Components various



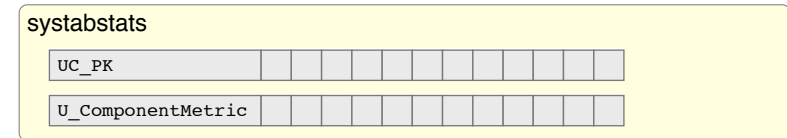
### DataStructure · Physical

Placement; etc



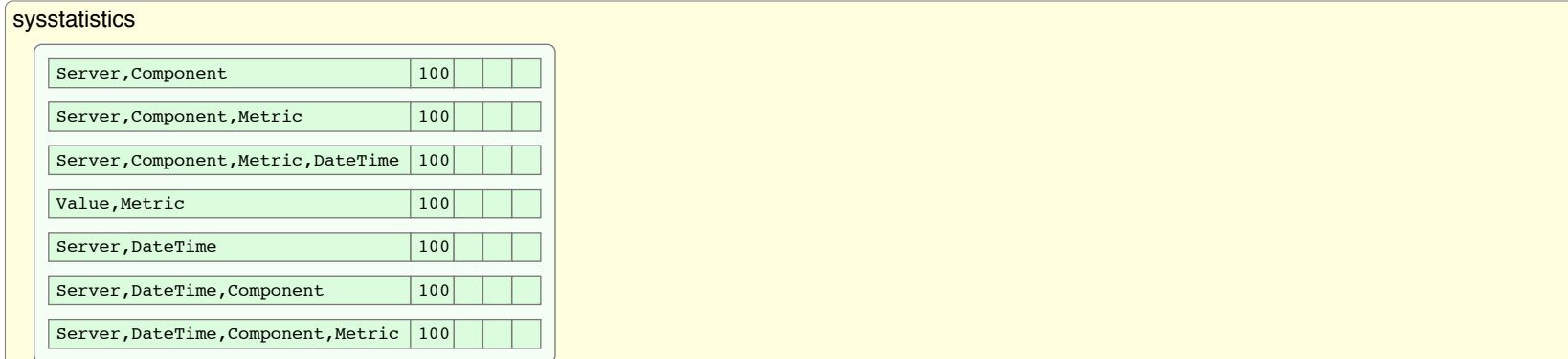
### Index Metrics

Height; RowSize; Row, Page, Extent & Cluster Ratio counts



### ColumnGroup Stats

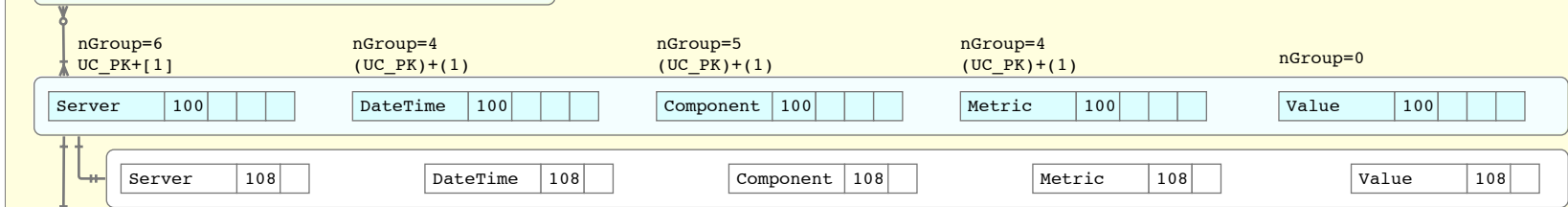
ColumnGroups Density & Selectivity



### Column Stats

Density & Selectivity

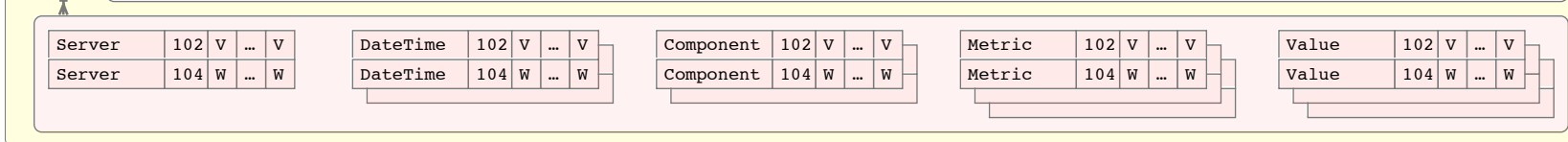
DataChange counter



### Histogram

Cell Value

Cell Weight



The different UPDATE STATISTICS operations, and how they affect the relevant sysstatistics rows, are illustrated on the following pages. The examples are given in a sequence, not in isolation from each other.

# Sybase Statistics Demystified

## 4.2 Update Stats • Column



### UPDATE STATISTICS *Table (Column)*

- Full Stats (Densities and an Histogram) can be created for any column

Server,Component	100			
Server,Component,Metric	100			
Server,Component,Metric,DateTime	100			
Value,Metric	100			
Server,DateTime	100			
Server,DateTime,Component	100			
Server,DateTime,Component,Metric	100			

Server	100				DateTime	100				Component	100				Metric	100				Value	100			
--------	-----	--	--	--	----------	-----	--	--	--	-----------	-----	--	--	--	--------	-----	--	--	--	-------	-----	--	--	--

Server	102	V	...	V	DateTime	102	V	...	V	Component	102	V	...	V	Metric	102	V	...	V	Value	102	V	...	V
Server	104	W	...	W	DateTime	104	W	...	W	Component	104	W	...	W	Metric	104	W	...	W	Value	104	W	...	W

SM\_ComponentValue

If an Index with the column as a leading column exists, it is read (Leaf Level via the PageChain). Otherwise ASE has to access the data:

- APL CI: the Clustered Index (Leaf Level via the PageChain)
- APL Heap: via the PageChain
- DOL: the Heap via OAM Pages

4.2

```
UPDATE STATISTICS SM_ComponentValue (Value)
1 Creates/updates Column Stats for the column
2 Creates/updates Histogram for the column
```

UC\_PK ( Server, DateTime, Component, Metric )

- Column requested is Unindexed
- CI Leaf Level used

#### Resource Use

- The leading column in an Index (if one can be used) is already sorted.
- Non-leading columns need to be sorted: that requires a worktable in tempdb, and sort buffers in the procedure cache.

#### Sampling

WITH SAMPLING = *Percent*

- applies to non-leading and unindexed Column
- does not update Column & ColumnGroup Stats (updates Histogram only)
- accesses the data rows (leaf-level of CI for APL; Heap for DOL).

#### Transaction Isolation

UPDATE STATISTICS uses:

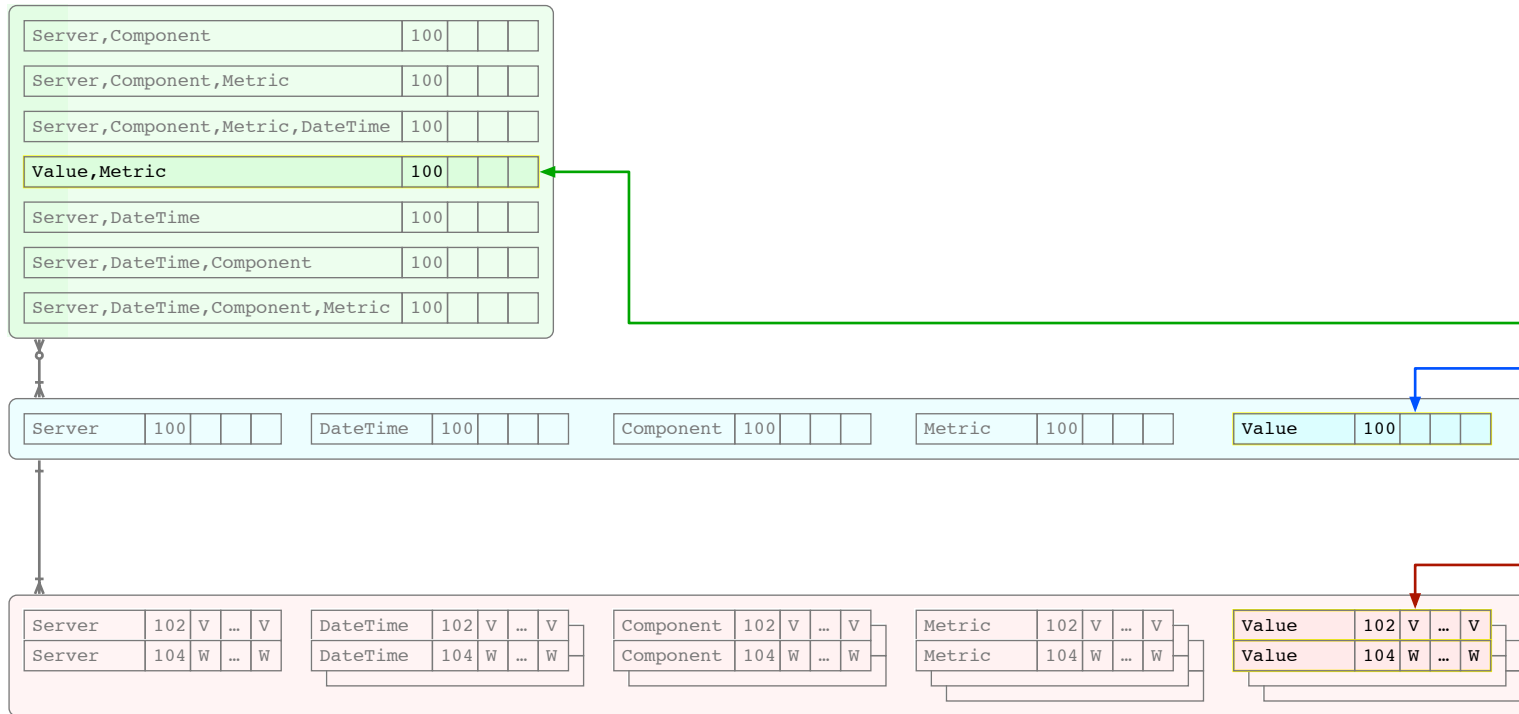
- APL: ISOLATION LEVEL 1 (READ COMMITTED)
- DOL: ISOLATION LEVEL 0 (READ UNCOMMITTED)

# 4.3 Update Stats • ColumnGroup



## UPDATE STATISTICS *Table* (*Col\_1*, *Col\_2*, ...)

- Any list of columns identified together and given to UPDATE STATISTICS in one command, forms a ColumnGroup
- Obviously, it is only required where there is no Index for the ColumnGroup  
Where there is an index, use UPDATE INDEX STATISTICS *Table* [*Index*] (next page).



SM\_ComponentValue

If an Index with the ColumnGroup exists, it is read (Leaf Level via the PageChain). Otherwise ASE has to access the data:

- APL CI: the Clustered Index (Leaf Level via the PageChain)
- APL Heap: via the PageChain
- DOL: the Heap via OAM Pages

4.3

UPDATE STATISTICS SM\_ComponentValue (Value, Metric)

- 1 Creates/updates ColumnGroup Stats for the second and subsequent increment in the group
- 2 Creates/updates Column Stats for the leading Column
- 3 Creates/updates Histogram for the leading Column

UC\_PK ( Server, DateTime, Component, Metric )

- ColumnGroup requested is Unindexed
- CI Leaf Level used

**Record Filing System**

- The "Primary Keys" are surrogates (Record IDs; etc), not Relational Keys
- They tend to have single-column indices, and many more of them
- All except the "Primary Key" index are non-unique
- The value of ColumnGroups is lost: ColumnGroups must be added manually
- Statistics need to be updated frequently

**Relational Database**

- On the other hand, these have genuine Relational Keys, which are compound keys, and fewer indices overall
- The indices are mostly unique, or close to unique
- ColumnGroups are heavily used, and used directly, from the Indices (next page)
- Once the data distribution settles down, the statistics (all of them) settle down as well, and thus need to be updated less frequently.

**Resource Use**

- The leading column in an Index (if one can be used) is already sorted.
- Non-leading columns need to be sorted: that requires a worktable in tempdb, and sort buffers in the procedure cache.

**Sampling**

WITH SAMPLING = *Percent*

- applies to non-leading and unindexed Column
- does not update Column & ColumnGroup Stats (updates Histogram only)
- accesses the data rows (leaf-level of CI for APL; Heap for DOL).

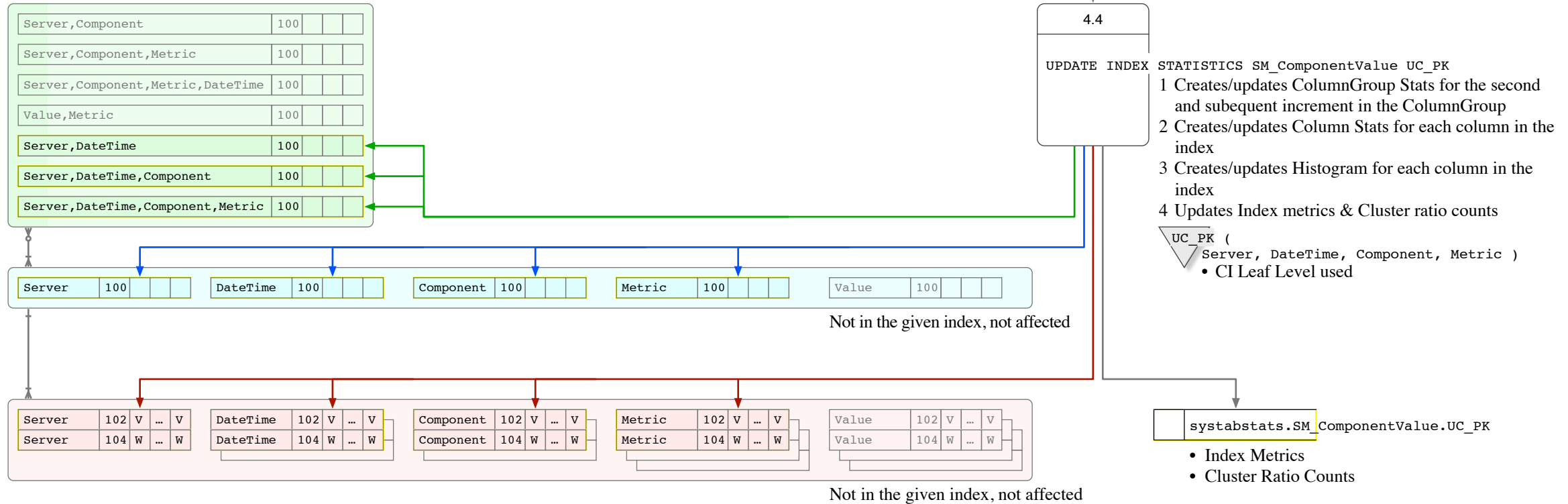
**Transaction Isolation**

UPDATE STATISTICS uses:

- APL: ISOLATION LEVEL 1 (READ COMMITTED)
- DOL: ISOLATION LEVEL 0 (READ UNCOMMITTED)

### UPDATE INDEX STATISTICS *Table [Index]*

- Takes second and subsequent columns in the Index as ColumnGroups, and generates Density Stats for them
- Generates Density Stats and an Histogram for each column in the Index
- If the Index parameter is omitted, all indices for the table are processed, without doubling up at the column level



**Second Index**

Where performance and the maintenance window are a consideration, particularly when the number of Cells is large, a duplication of effort with no gain, that is redundant, is to be avoided. Note that at this point in the sequence for the example, it would be very inefficient to update Stats for the second index, issuing:

```
UPDATE INDEX STATISTICS SM_ComponentValue SM_ComponentValue
```

because the Column Stats for four of the five columns in that index have just been freshly updated, and that command would cause them to be updated again. Instead, update the Stats for the remaining ColumnGroups only, via:

```
UPDATE STATISTICS SM_ComponentValue (Server, Component, Metric, Datetime)
```

**Resource Use**

- The leading column in an Index (if one can be used) is already sorted.
- Non-leading columns need to be sorted: that requires a worktable in tempdb, and sort buffers in the procedure cache.

**Sampling**

WITH SAMPLING = *Percent*

- applies to non-leading and unindexed Column
- does not update Column & ColumnGroup Stats (updates Histogram only)
- accesses the data rows (leaf-level of CI for APL; Heap for DOL).

**Transaction Isolation**

UPDATE STATISTICS uses:

- APL: ISOLATION LEVEL 1 (READ COMMITTED)
- DOL: ISOLATION LEVEL 0 (READ UNCOMMITTED)

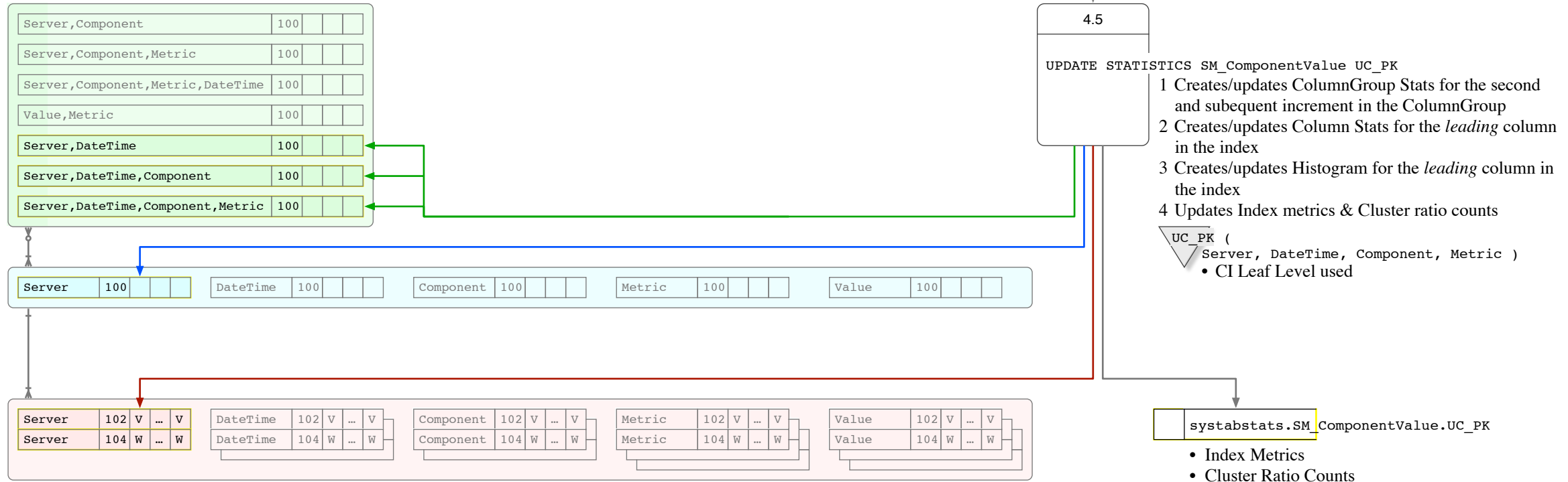
# 4.5 Update Stats • Index (Deprecated)

## UPDATE STATISTICS *Table* [*Index*]

- The older, deprecated form of the command, If Index is omitted, all indices for the table are processed

## CREATE ... INDEX *Table* ( *Col\_1* [, *Col\_2...*] )

- Automatically executes the older form, if the table is populated.
- Therefore, ensure that UPDATE INDEX STATISTICS is executed after populating a new table.



**Resource Use**

- The leading column in an Index (if one can be used) is already sorted.
- Non-leading columns need to be sorted: that requires a worktable in tempdb, and sort buffers in the procedure cache.

**Sampling**

WITH SAMPLING = *Percent*

- applies to non-leading and unindexed Column
- does not update Column & ColumnGroup Stats (updates Histogram only)
- accesses the data rows (leaf-level of CI for APL; Heap for DOL).

**Transaction Isolation**

UPDATE STATISTICS uses:

- APL: ISOLATION LEVEL 1 (READ COMMITTED)
- DOL: ISOLATION LEVEL 0 (READ UNCOMMITTED)



# Sybase Statistics Demystified

## 4.6 Update Stats • Table

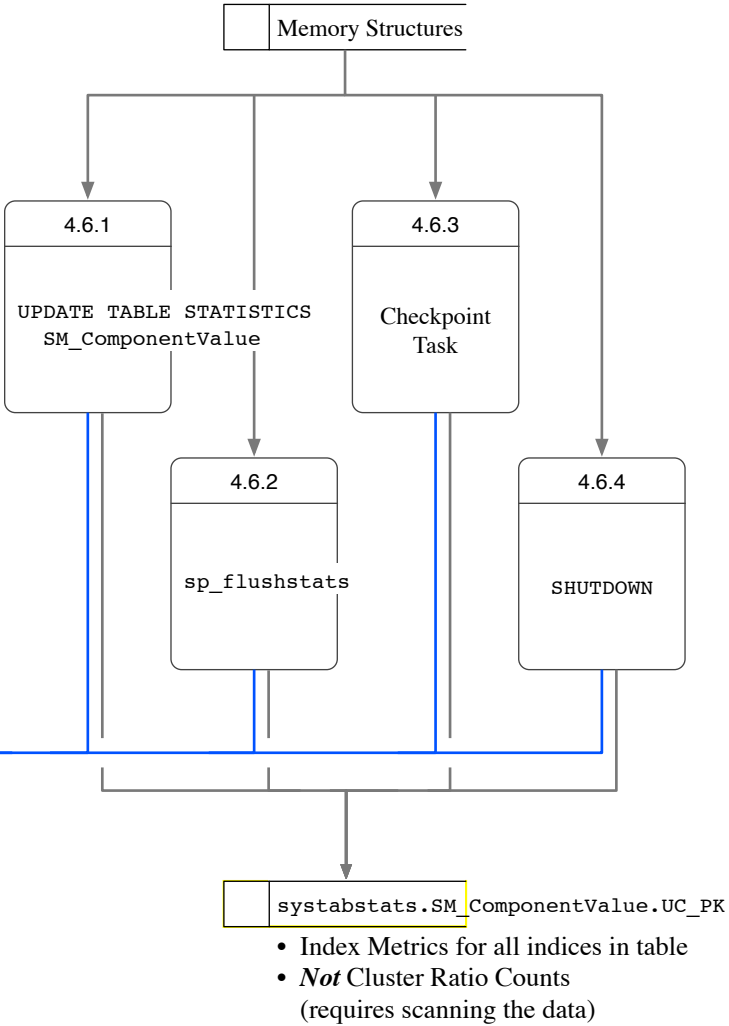
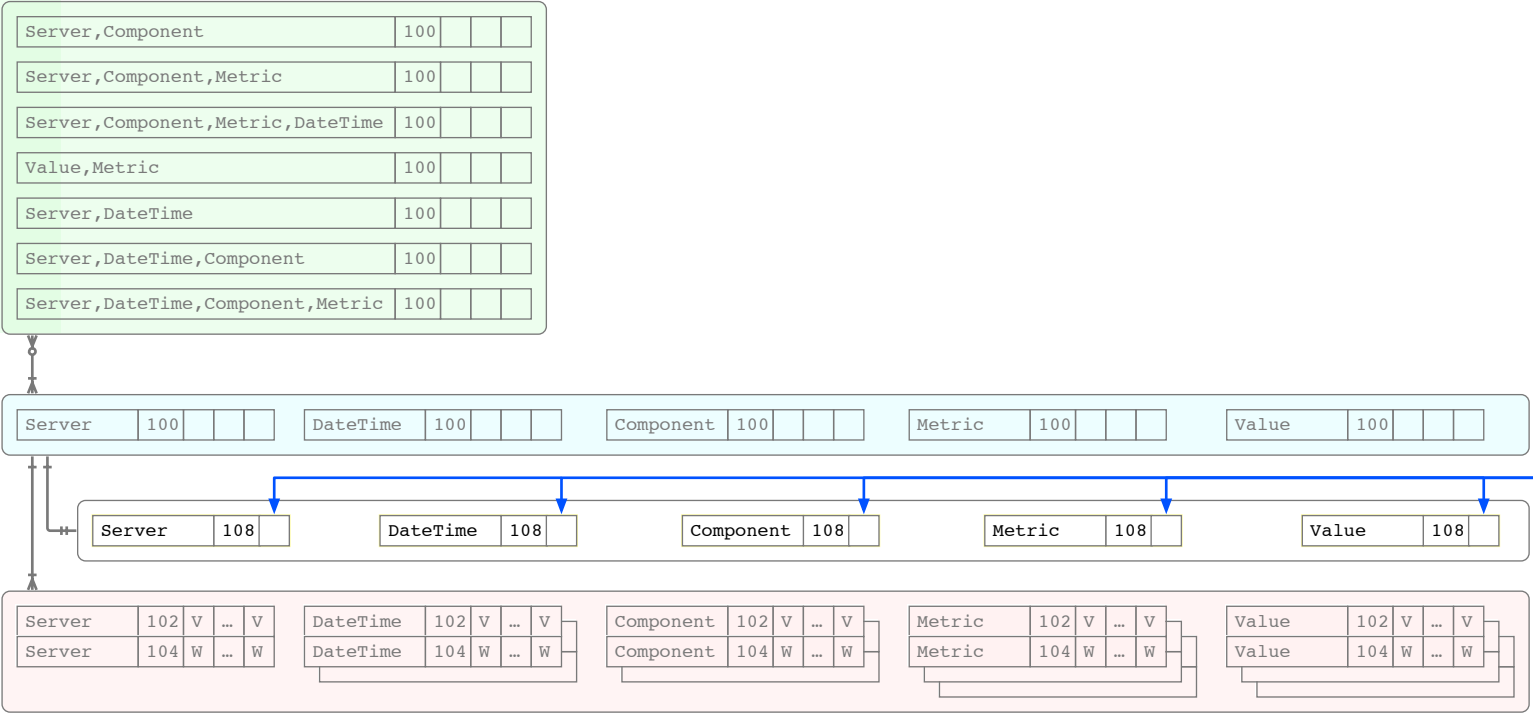


### UPDATE TABLE STATISTICS *Table*

- Update Table Stats is pretty much obsolete.

### EXEC sp\_flushstats *Table*

- The CHECKPOINT task updates systabstats and the DataChange counters periodically, from ASE memory structures; the frequency is based on recovery interval.
- SHUTDOWN updates it as well (WITH NOWAIT does not, of course).
- Obviously sp\_flushstats on a single table is much faster than CHECKPOINT or SHUTDOWN.



**Resource Use**

- The leading column in an Index (if one can be used) is already sorted.
- Non-leading columns need to be sorted: that requires a worktable in tempdb, and sort buffers in the procedure cache.

**Sampling**

WITH SAMPLING = *Percent*

- applies to non-leading and unindexed Column
- does not update Column & ColumnGroup Stats (updates Histogram only)
- accesses the data rows (leaf-level of CI for APL; Heap for DOL).

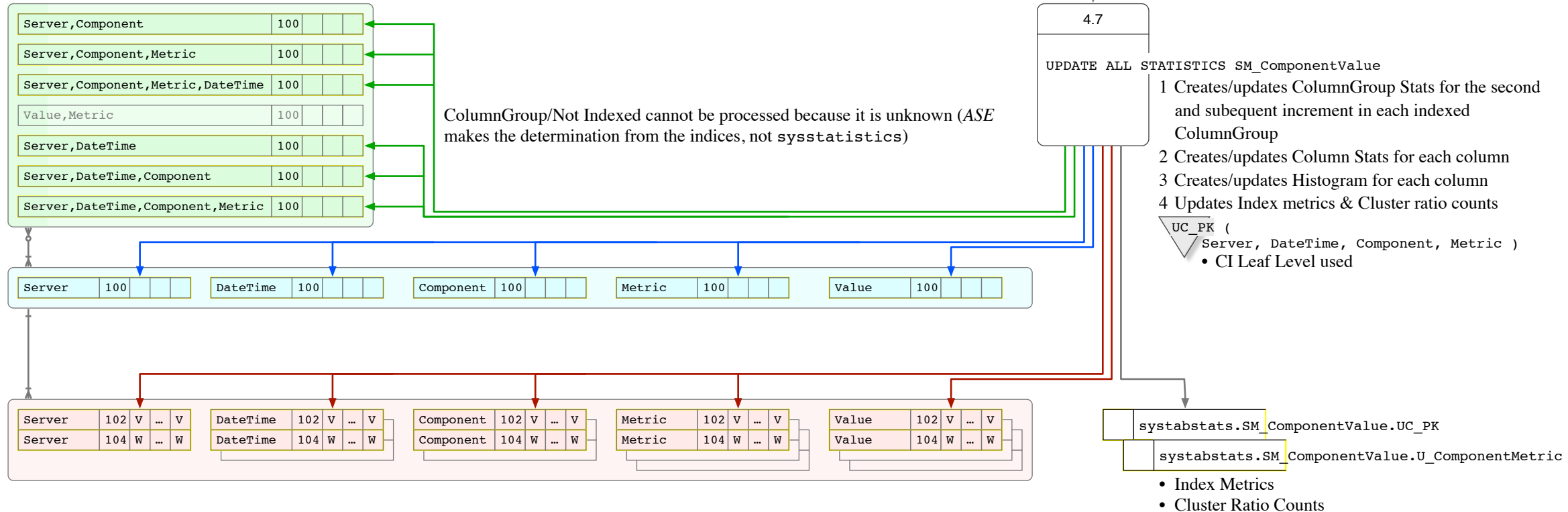
**Transaction Isolation**

UPDATE STATISTICS uses:

- APL: ISOLATION LEVEL 1 (READ COMMITTED)
- DOL: ISOLATION LEVEL 0 (READ UNCOMMITTED)

### UPDATE ALL STATISTICS *Table*

- Generates Density Stats and an Histogram for all columns, whether indexed or not
- Takes all columns in each Index as a ColumnGroup, and generates Density Stats for them
- ColumnGroups/Not Indexed cannot be processed because they are unknown
- Generally, it is overkill, the last resort for people who refuse to learn about Stats administration. Generating, and maintaining, Stats for columns that are not used in queries is a waste.
- Determining the specific problem as to which Stats are not being updated, and why, is far superior.



**Resource Use**

- The leading column in an Index (if one can be used) is already sorted.
- Non-leading columns need to be sorted: that requires a worktable in tempdb, and sort buffers in the procedure cache.

**Sampling**

WITH SAMPLING = *Percent*

- applies to non-leading and unindexed Column
- does not update Column & ColumnGroup Stats (updates Histogram only)
- accesses the data rows (leaf-level of CI for APL; Heap for DOL).

**Transaction Isolation**

UPDATE STATISTICS uses:

- APL: ISOLATION LEVEL 1 (READ COMMITTED)
- DOL: ISOLATION LEVEL 0 (READ UNCOMMITTED)

# 5 Goal



Customers only

Customers only

### Help Statistic

Both Column Stats and ColumnGroup Stats, that are reported in `optdiag`, plus a number of additional metrics which need to be considered together, are reported in Software Gems' *HelpStatistic* report, horizontally, and in full context. The Histogram (per column) is reported via *HelpHistogram*. The reports are explained in the next few pages. This page illustrates the overall concept, and the `optdiag` reports that are related.

Table/ColumnGroup/Column	nGrp	Index/Notice	Un	M	nUpd	dUpdated	nReq	nAct	rHist	Delta	Density_Range	Density_Total	Uniqueness	nValue_Unique	Value_Range	Value_Total
SM_ComponentValue	[2]		[2]						16					5737		
Server,Component,Metric,DateTime		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.0001743071	1.0	5737	0.0	0.000174307
Server,Component,Metric		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.0092919949	0.01987	114	0.0	0.00877193
Server,Component		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.1025204531	0.00209	12	0.0	0.083333333
Server	6	UC_PK+[1]			3	11/08/26 02:03	20	2	2	0.00	0.0	1.0	0.00017	1	0.0	1.0
DateTime	4	(UC_PK)+(1)			3	11/08/26 01:50	100	192	6	0.00	0.0	0.0149656510	0.01673	96	0.0	0.010416667

### Help Histogram

Table	Column	Cell	Weight	Op	Value
SM_ComponentValue	DateTime	1	0	<	
		2	0.1246290803	=	10/12/21 05:00
		3	0	<	
		4	0.1246290803	=	10/12/21 12:00
		5	0	<	
		6	0.0652818978	=	10/12/21 18:00
		7	0	<	
		8	0.0652818978	=	10/12/21 22:00
		9	0	<	
		10	0.1246290803	=	11/02/07 03:00
		11	0	<	
		12	0.0474777445	=	11/02/07 04:00
		13	0.0118694361	<=	11/02/07 05:00
		14	0	<	
		15	0.0623145401	=	11/02/07 09:00
		16	0	<	
		17	0.1246290803	=	11/02/07 11:00
		18	0	<	
		19	0.0178041551	=	11/02/07 15:00
		20	0	<	
		21	0.0415430255	=	11/02/07 16:00
		22	0	<	
		23	0.0652818978	=	11/02/07 18:00
		24	0.0118694361	<=	11/02/14 11:00
		25	0	<	
		26	0.0623145401	=	11/02/14 12:00
		27	0	<	
		28	0.0504451022	=	11/02/14 13:00

### optdiag

```

Table owner: "dbo"
Table name: "SM_ComponentValue"

Statistics for column: "DateTime"
Last update of column statistics: Aug 11 2011
11:08:31:346AM

Range cell density: 0.0118694362017804
Total density: 0.0898044360697021
Range selectivity: default used (0.33)
In between selectivity: default used (0.25)
Unique range values: 0.0118694362017804
Unique total values: 0.0666666666666667
Average column width: default used (4.00)

Histogram for column: "DateTime"
Column datatype: smalldatetime
Requested step count: 20
Actual step count: 28
Sampling Percent: 10
Out of range Histogram Adjustment is ON.

Step Weight Value
1 0.00000000 < "Dec 21 2010 5:00AM"
2 0.12462908 = "Dec 21 2010 5:00AM"
3 0.00000000 < "Dec 21 2010 12:00PM"
4 0.12462908 = "Dec 21 2010 12:00PM"
5 0.00000000 < "Dec 21 2010 6:00PM"
6 0.06528190 = "Dec 21 2010 6:00PM"
7 0.00000000 < "Dec 21 2010 10:00PM"
8 0.06528190 = "Dec 21 2010 10:00PM"
9 0.00000000 < "Feb 7 2011 3:00AM"
10 0.12462908 = "Feb 7 2011 3:00AM"
11 0.00000000 < "Feb 7 2011 4:00AM"
12 0.04747774 = "Feb 7 2011 4:00AM"
13 0.01186944 <= "Feb 7 2011 5:00AM"
14 0.00000000 < "Feb 7 2011 9:00AM"
15 0.06231454 = "Feb 7 2011 9:00AM"
16 0.00000000 < "Feb 7 2011 11:00AM"
17 0.12462908 = "Feb 7 2011 11:00AM"
18 0.00000000 < "Feb 7 2011 3:00PM"
19 0.01780416 = "Feb 7 2011 3:00PM"
20 0.00000000 < "Feb 7 2011 4:00PM"
21 0.04154303 = "Feb 7 2011 4:00PM"
22 0.00000000 < "Feb 7 2011 6:00PM"
23 0.06528190 = "Feb 7 2011 6:00PM"
24 0.01186944 <= "Feb 14 2011 11:00AM"
25 0.00000000 < "Feb 14 2011 12:00PM"
26 0.06231454 = "Feb 14 2011 12:00PM"
27 0.00000000 < "Feb 14 2011 1:00PM"
28 0.05044510 = "Feb 14 2011 1:00PM"
    
```

This shows example reports from `optdiag`:

- Density and Selectivity Stats for a ColumnGroup (green)
- Density and Selectivity Stats for a Column (blue)
- Histogram (distribution) for a Column (pink)

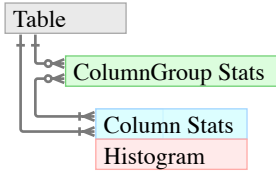
```

Statistics for column group: "Server", "Component"
Last update of column statistics: Aug 12 2011
7:14:13:426PM

Range cell density: 0.0000000000000000
Total density: 0.1025204531353392
Range selectivity: default used (0.33)
In between selectivity: default used (0.25)
Unique range values: 0.0000000000000000
Unique total values: 0.0833333333333333
Average column width: default used (3.00)
    
```

Software Gems' *HelpHistogram* report may be easier to comprehend than the `optdiag` report. The `ShowPhysical` option matches the format of `optdiag` (left) in order that it can be readily compared and verified

- `optdiag` rows and columns are maintained
- Range** Cells ("`<=`" in `optdiag`) are unchanged [2]
- Placeholder** for Frequency Cells ("`<`") are shown without a Weight or Value (because they have none)
- Frequency** Cells "`=`" are unchanged [13]
- Both Range and Frequency Cells are more obvious, and ones that are expected but missing stand out.



# 6.2 Help Statistic



## 6.2.1 Help Statistic

This page describes the content of the *Software Gems' HelpStatistic* report. In its default operation, Columns and ColumnGroups Stats are reported at the table level, partition level Stats are not shown. The report shows all Column and ColumnGroup Stats for the tables requested; and allows columns to be easily related to their ColumnGroups and Indices. The Histograms can be viewed via **HelpHistogram** <sup>13</sup>.

**Table entries** are useful in organising the **HelpStatistic** report. However, except for the first column, they would be empty. Therefore the following columns are used (overloaded) with additional (non-Stats) information which is typically sought when viewing the report, saving a reference to the **HelpIndex** or other report:

Grp [number of indices]  
 Un [number of unique indices]  
 nValue\_Unique number of rows in the table

**Table/ColumnGroup/Column**

The main identifier in the report, each entry represents either a Table, or a ColumnGroup Stat, or a Column Stat, that has been stored. For ease of reference, the entry is ordered by:

Table Table entry  
 ColumnGroup ColumnGroup entry  
 Column Column entry

The number of **ColumnGroups** this column appears in. ColumnGroups are readily identifiable from the few lines immediately above

**Index**

**ColumnGroup entry:**

Index the first Index in which the ColumnGroup is indexed (matches the leading columns). If there is more than one, the remainder are identified thus: +[n] the ColumnGroup is indexed an additional n times (identifying possible duplicate indices)

**Column entry:**

Index the first Index in which the column is indexed (matches the leading column). If there is more than one, the remainder are identified thus: +[n] the column is indexed an additional n times (identifying possible duplicate indices)

(Index) the first Index in which the column appears as a non-leading column. If there is more than one, the remainder are identified thus: + (n) the column appears in n additional indices

Indicates that ASE reported that Stats for the identified Column or Column Group are **Missing** (if such is being captured). This treatment presents it in the context of the rest of the Stats, and thus allows considered evaluation, that would otherwise be isolated. nUpd is the no of occasions the Stats were found to be missing (low occurrences can be safely ignored). The Missing count is particularly useless in the absence of used count for Stats, as there is nothing to compare against, but it does provide a flag for sites that do not administer Stats.

The number of **Requested & Actual Cells** in the Histograms for the column

The number of rows (must be at least 2) used to store the **Histograms** for the column or table

**Data Change** in the column since Stats were last updated (constrained to 999 for sanity)

Very important single indicator, even for non-distinct columns, once it is understood

Distinct values for the column or ColumnGroup determined by Stats

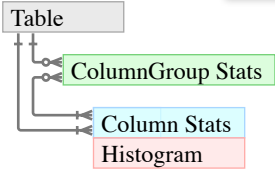
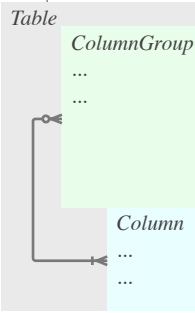
Shown only if they differ from Density

Index is Unique

The number of times this Statistic has been updated, and datetime of the last update

The **Density** (Statistics) for the Column or a ColumnGroup.

Table/ColumnGroup/Column	nGrp	Index/Notice	Un	M	nUpd	dUpdated	nReq	nAct	rHist	Delta	Density_Range	Density_Total	Uniqueness	nValue_Unique	Value_Range	Value_Total
C_AccessSequence	[1]	-NoGrpStats-	[1]						4					1		
SubjectAreaCode	0	UC_Base	Un		3	10/01/06 23:10	20	2	2	0.00	1.0	1.0	1.0	1	1.0	1.0
Sequence	0	(UC_Base)	Un		3	10/01/06 23:10	20	2	2	0.00	1.0	1.0	1.0	1	1.0	1.0
SM_ComponentMetric	[1]	-NoStats-	[1]						0					122		
SM_ComponentValue	[2]		[2]						16					5737		
Server,DateTime,Component,Metric		UC_PK	Un		3	11/08/26 02:01					0.0	0.0001743071	1.0	5737	0.0	0.000174307
Server,Component,Metric,DateTime		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.0001743071	1.0	5737	0.0	0.000174307
Server,Component,Metric		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.0092919949	0.01987	114	0.0	0.00877193
Server,DateTime,Component		UC_PK	Un		3	11/08/26 02:01					0.0	0.0020003847	0.11713	672	0.0	0.001488095
Server,DateTime		UC_PK	Un		3	11/08/26 02:01					0.0	0.0149656510	0.01673	96	0.0	0.010416667
Server,Component		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.1025204531	0.00209	12	0.0	0.083333333
Component,Metric					3	11/08/26 02:09					0.0	0.0092919949	0.01987	114	0.0	0.00877193
Server	6	UC_PK+[1]			3	11/08/26 02:03	20	2	2	0.00	0.0	1.0	0.00017	1	0.0	1.0
DateTime	4	(UC_PK)+(1)			3	11/08/26 01:50	100	192	6	0.00	0.0	0.0149656510	0.01673	96	0.0	0.010416667
Component	6	(UC_PK)+(1)			3	11/08/26 02:09	300	24	2	0.00	0.0	0.1025204531	0.00209	12	0.0	0.083333333
Metric	4	(UC_PK)+(1)			3	11/08/26 01:59	300	223	6	0.00	0.0001743071	0.0095720044	0.01952	112	0.000174307	0.008928571



**Notice**

One of the following is shown for Table entries, in order of importance (ie. if a Notice is shown, any Notices below its importance will not be shown):

- NoStats- there are no Stats for anything in the table
- NoColStats- there are no Stats for Columns in the table
- NoGrpStats- there are no Stats for ColumnGroups in the table
- NoCells- there are no Cells (Histograms) for any columns in the table

**ColumnGroup entries:**

-MissingGrpStats- ASE has recorded that Stats for the ColumnGroup were sought, and not found

**Column entries:**

-MissingColStats- ASE has recorded that Stats for the Column were sought, and not found

<sup>13</sup> For detailed information in other areas, refer to the **HelpTable**, **HelpIndex** (derived stats and index metrics belong with the index) or **HelpPartition** (partition level) reports.



6.2.2 Help Statistic/Partition

In its default execution, the **HelpStatistic** report provides information on Stats at the table level. For partitioned tables, *Sybase automatically* stores ColumnGroup and Column Stats (including the Histograms) for each partition. For administrators who manage their Stats at the partition level, the **ShowPartition** option provides such a report, with additional columns for the partition level information. Only the rows and columns that are *additional* (to the table level report on the previous page) are described here.

Regarding partition level Stats, the concept is, the Column or ColumnGroup exists in each partition, therefore in logical terms, partition is *below* Column or ColumnGroup. This is consistent with the optdiag report.

*Table entries are useful in organising the HelpStatistic report. However, except for the first column, they would be empty. Therefore the following columns are used (overloaded) with additional (non-Stats) information which is typically sought when viewing the report, saving a reference to the HelpIndex or other report:*

- Grp [number of indices]
- Un [number of unique indices]
- nReq [number of partitions]

**Table/ColumnGroup/Column**

The main identifier in the report, each entry represents either a Table, or a ColumnGroup Stat, or a Column Stat, that has been stored. For ease of reference, the entry is ordered by:

- Table
- ColumnGroup
- ColumnGroup.Partition
- Column
- Column.Partition

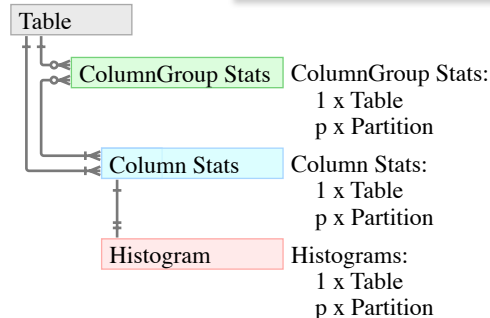
The number of **Requested & Actual Cells** in the Histograms for the column

The number of rows used to store the **table level** Histogram for the column

The number of rows used to store the **partition level** Histograms for the column

The row count for the table (table entry) or Partition (partition entry). This should be evaluated against ValueUnique determined by Stats

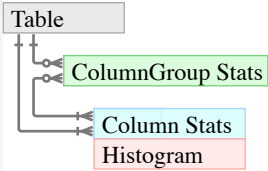
	Table/ColumnGroup/Column.Parttn	nGrp	Index/Notice	Un	M	nUpd	dUpdated	nReq	nAct	rH_Tbl	rH_Ptn	Delta	Density_Range	Density_Total	Uniqueness	nRow	nValue_Unique	Value_Range	Value_Total
Table	TestBase_APL	[2]		[2]				[4]		63	224					2000010			
ColumnGroup	SecurityId,Name					3	11/08/22 00:00						0.0000005000	0.0000005000	1.0	2000010	2000010	0.0000005000	0.0000005000
Partition	SecurityId,Name.1536158					3	11/08/21 23:59						0.0000040256	0.0000040256	0.50000	496821	248410	0.0000040256	0.0000040256
	SecurityId,Name.1169540319					3	11/08/21 23:59						0.0000040258	0.0000040258	0.50061	496195	248400	0.0000040258	0.0000040258
	SecurityId,Name.1185540376					3	11/08/22 00:00						0.0000040258	0.0000040258	0.50071	496091	248400	0.0000040258	0.0000040258
	SecurityId,Name.1201540433					3	11/08/22 00:00						0.0000039169	0.0000039169	0.49971	510903	255303	0.0000039169	0.0000039169
	Name,SecurityId					3	11/08/21 23:58						0.0000005453	0.0000005453	0.91685	2000010	1833702	0.0000005453	0.0000005453
	Name,SecurityId.1536158					3	11/08/21 23:55						0.0000020128	0.0000020128	1.0	496821	496821	0.0000020128	0.0000020128
	Name,SecurityId.1169540319					3	11/08/21 23:56						0.0000020153	0.0000020153	1.0	496195	496195	0.0000020153	0.0000020153
	Name,SecurityId.1185540376					3	11/08/21 23:57						0.0000020158	0.0000020158	1.0	496091	496091	0.0000020158	0.0000020158
	Name,SecurityId.1201540433					3	11/08/21 23:58						0.0000019573	0.0000019573	1.0	510903	510903	0.0000019573	0.0000019573
Column	SecurityId	2 UC_SecurityId	Un			3	11/08/22 00:00	1000	1090	28	104	0.00	0.0000005000	0.0000005000	1.0	2000010	2000010	0.0000005000	0.0000005000
Partition	SecurityId.1536158					3	11/08/21 23:59	1000	1000		26	0.00	0.0000020128	0.0000020128	1.0	496821	496821	0.0000020128	0.0000020128
	SecurityId.1169540319					3	11/08/21 23:59	1000	1000		26	0.00	0.0000020153	0.0000020153	1.0	496195	496195	0.0000020153	0.0000020153
	SecurityId.1185540376					3	11/08/22 00:00	1000	1000		26	0.00	0.0000020158	0.0000020158	1.0	496091	496091	0.0000020158	0.0000020158
	SecurityId.1201540433					3	11/08/22 00:00	1000	1000		26	0.00	0.0000019573	0.0000019573	1.0	510903	510903	0.0000019573	0.0000019573
	Name	2 U_Name	Un			3	11/08/21 23:58	1000	1163	35	120	0.00	0.0000005453	0.0000005453	0.91685	2000010	1833702	0.0000005453	0.0000005453
	Name.1536158					3	11/08/21 23:55	1000	1000		30	0.00	0.0000020128	0.0000020128	1.0	496821	496821	0.0000020128	0.0000020128
	Name.1169540319					3	11/08/21 23:56	1000	1000		30	0.00	0.0000020153	0.0000020153	1.0	496195	496195	0.0000020153	0.0000020153
	Name.1185540376					3	11/08/21 23:57	1000	1000		30	0.00	0.0000020158	0.0000020158	1.0	496091	496091	0.0000020158	0.0000020158
	Name.1201540433					3	11/08/21 23:58	1000	1000		30	0.00	0.0000019573	0.0000019573	1.0	510903	510903	0.0000019573	0.0000019573



6.3.1 Introduction

Software Gems' *HelpStatistic* report should satisfy most Stats administration needs.

Table/ColumnGroup/Column	nGrp	Index/Notice	Un	M	nUpd	dUpdated	nReq	nAct	rHist	Delta	Density_Range	Density_Total	Uniqueness	nValue_Unique	Value_Range	Value_Total
SM_ComponentValue	[2]		[2]						16					5737		
Server,Component,Metric,DateTime		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.0001743071	1.0	5737	0.0	0.000174307
Server,Component,Metric		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.0092919949	0.01987	114	0.0	0.00877193
Server,Component		U_ComponentMetric	Un		3	11/08/26 02:03					0.0	0.1025204531	0.00209	12	0.0	0.083333333
Server	6	UC_PK+[1]			3	11/08/26 02:03	20	2	2	0.00	0.0	1.0	0.00017	1	0.0	1.0
DateTime	4	(UC_PK)+(1)			3	11/08/26 01:50	100	192	6	0.00	0.0	0.0149656510	0.01673	96	0.0	0.010416667



However there are times when the full extent of Statistics detail need to be inspected. The companion *HelpHistogram* report provides full details of the Histograms for the requested tables. Of course it can be related directly to the *optdiag* report. There are two report formats.

Table	Column	Cell	Weight	Op	Value
SM_ComponentValue	DateTime	1	0	<	
		2	0.1246290803	=	10/12/21 05:00
		3	0	<	
		4	0.1246290803	=	10/12/21 12:00
		5	0	<	
		6	0.0652818978	=	10/12/21 18:00
		7	0	<	
		8	0.0652818978	=	10/12/21 22:00
		9	0	<	
		10	0.1246290803	=	11/02/07 03:00
		11	0	<	
		12	0.0474777445	=	11/02/07 04:00
		13	0.0118694361	<=	11/02/07 05:00
		14	0	<	
		15	0.0623145401	=	11/02/07 09:00
		16	0	<	
		17	0.1246290803	=	11/02/07 11:00
		18	0	<	
		19	0.0178041551	=	11/02/07 15:00
		20	0	<	
		21	0.0415430255	=	11/02/07 16:00
		22	0	<	
		23	0.0652818978	=	11/02/07 18:00
		24	0.0118694361	<=	11/02/14 11:00
		25	0	<	
		26	0.0623145401	=	11/02/14 12:00
		27	0	<	
		28	0.0504451022	=	11/02/14 13:00

Table	Column	Cell	F	Value	Pct
SM_ComponentValue	DateTime	1		-Null-	
		2	F	10/12/21 05:00	12.46%
		3	F	10/12/21 12:00	12.46%
		4	F	10/12/21 18:00	6.53%
		5	F	10/12/21 22:00	6.53%
		6	F	11/02/07 03:00	12.46%
		7	F	11/02/07 04:00	4.75%
		8		11/02/07 05:00	1.19%
		9	F	11/02/07 09:00	6.23%
		10	F	11/02/07 11:00	12.46%
		11	F	11/02/07 15:00	1.78%
		12	F	11/02/07 16:00	4.15%
		13	F	11/02/07 18:00	6.53%
		14		11/02/14 11:00	1.19%
		15	F	11/02/14 12:00	6.23%
		16	F	11/02/14 13:00	5.04%

The default or **ShowLogical** report.

- It shows the logical Cells that we need to be concerned with (the physical Cells per *optdiag*. are on the left)
- Placeholder Cells are eliminated: 16 rows instead of 28
- The Null Cell is shown as Null
- F differentiates Frequency Cells [13] from Range Cells [3], the distribution of which happen to be small
- Weight is shown as a percentage, which most people readily understand.

6.3.2 Advantage over *optdiag*

Rather than an utility program, *HelpHistogram* is provided as a stored procedure, which makes it substantially easier to access and use. The report is therefore a result set, which means the data is also easier to access; use; transfer to a spreadsheet; erect a chart; etc.

- The improved readability is afforded by:
- Elimination of the Placeholder Cells, resulting in a shorter report
  - trailing zeroes for values of 0 and 1 in the Weight are truncated (or, zero-fill removed):
    - the empty space makes the ordinary Weight and Values stand out; the report is otherwise a fat rectangular block of unreadable numbers
  - Range and Frequency Cells [F] are therefore more distinct, differentiated
  - the Range Cell for Nulls is identified as such (the confusing value eliminated), and differentiated from a minimum Value cell that may exist
  - Cell Weight is displayed as a percentage

- This allows:
- quick problem identification
  - quick and *accurate* problem diagnosis, eliminating errors, reversals, and wasted effort
  - the minimum entries confirming or denying the Stats that are required to be easily identified, and executed, therefore minimising resolution time, as well as the effect on production

In order to allow verification of the reported Histogram, the **ShowPhysical** option matches the *optdiag* report: although the *optdiag* rows and columns (and silliness) are replicated for ready comparison, it does have improved readability within that scope.

The comparison of *optdiag* vs the *HelpHistogram/ShowPhysical* is in section [6.1].

**Sparse vs Dense**

A common error is that the administrator determines that the UPDATE STATS function does not work correctly, or at least that it is less than perfect. Essentially, it does not determine Sparse vs Dense Values the way a human would. This example provides an instance where such issues can be ventilated.

- The DateTime column has been determined as Sparse, quite correctly (where minutes are given, there is a large gap between hours), and Frequency Cells have been generated
- But a human might determine that the Values are Dense, on the basis that only the hours are given, and those hours are consecutive (which is incorrect, because minutes are given in the data)
- Thus he may speculate that Range Cells (less precise, and less Cells) would have been adequate

This issue needs to be recognised, in order to understand: why ASE generates the Histograms that it does; and that cursory examination of such issues is not adequate in solving problems. Technical endeavours demand absolute precision.

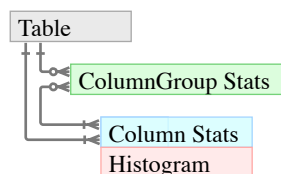
## 6.3 Help Histogram • Skew Value

### 6.3.3 Skewed Value

This report shows the Histograms for the `value` column in the `SM_ComponentValue` table:

- the Stats are beautifully, evenly, distributed in 27 Range Cells; they are quite adequate
- one Frequency Cell (Single Value Cell) for zero at 7.42% has been determined correctly; it is an example of a data skew away from the 2.97% norm
- in this instance, no work is required.

Table	Column	Cell	F	Value	Pct
SM_ComponentValue	Value	1		-Null-	
		2	F	0	7.42%
		3		3	4.15%
		4		13	2.97%
		5		37	2.97%
		6		62	3.56%
		7		143	2.97%
		8		258	2.97%
		9		508	2.97%
		10		883	2.97%
		11		2470	2.97%
		12		4104	2.97%
		13		5755	2.97%
		14		16600	2.97%
		15		22140	2.97%
		16		32560	2.97%
		17		44781	2.97%
		18		65214	2.97%
		19		83090	2.97%
		20		114152	2.97%
		21		150749	2.97%
		22		209891	2.97%
		23		288532	2.97%
		24		466617	2.97%
		25		611840	2.97%
		26		729180	2.97%
		27		1302682	2.97%
		28		1793188	2.97%



- In the instance where the data skew has not shown up as a Frequency Cell (Single Value Cell), use:

```
UPDATE STATS USING n VALUES
WITH SAMPLING = p
```

with  $n$  somewhat greater than the current number of Cells, and  $p$  a percentage that is adequate.

#### Sparse vs Dense

- The `value` column has been determined as Dense, quite correctly, and Range Cells have been generated
- The Values are not consecutive, hence one might determine they are Sparse. But since they are *all* sparse, the set (taken together) is in fact Dense
- As per the distribution (`Pct` column) is perfect for the range of values divided by 27.

# 6.3 Help Histogram • Repeated Value



## 6.3.4 Repeated Value

This section deals with highly repeated Values in a data Column.

This report shows the Histograms for the Metric column, before correction:

Table	Column	Cell	F	Value	Pct
SM_ComponentValue	Metric	1		-Null-	
		2		APF Req/Found Sp	3.56%
		3		Active Page	3.26%
		4		Buffer Washed	3.86%
		5		Byte Received	3.56%
		6		CPU Idle	3.56%
		7		Cache Miss	3.56%
		8		Context Switched	3.26%
		9		Device Interrupt	3.56%
		10		File/Page In	3.56%
		11		Flush/Commit	2.67%
		12	F	Flush/DOL Conflict	1.48%
		13		Flush/Single LR	4.15%
		14		Garbage Collected	3.86%
		15		IO Checked	4.45%
		16		IO Completed	3.26%
		17		Ins/APL Heap	3.56%
		18		LIO Req/Denied Pool	3.26%
		19		Log Alloc	3.56%
		20		Log Write	3.56%
		21		Network Other	3.26%
		22		PIO Queue	2.97%
		23		Priority Changed	3.26%
		24		Replacement/Scanned	3.26%
		25		Row/Updated	3.56%

- This is the Histogram for the Metric column produced USING 30 VALUES.
- In the first instance, the Stats appear to be adequate: the distribution appears to be close to even; the requested 30 VALUES produced 24 Range Cells & 1 Frequency Cell (Single Value Cell).
- The column is a Foreign Key to a Reference or Lookup table, wherein it has 112 distinct Values. The lookup Values are all used in the column, and used frequently in queries.
- Upon examination, it was determined that many of those Values are repeated in 5737 rows in the subject table.
- Let's say we are expecting millions of rows. Therefore while 24 Range Cells and 1 Frequency Cell covering 112 Values may be adequate, it is nowhere near optimal.
- The VALUES could be increased to say, 100, which would produce a better distribution, while remaining with Range Cells. Or the VALUES could be increased to 200, which would request one Frequency Cell (Single Value Cell) per lookup Value.

**Sparse vs Dense**

- For the Metric column, due to VALUES that were given as 30, Frequency Cells (Single Value Cells) are not possible, and Range Cells have been generated
- The Values are not consecutive, hence one might determine they are Sparse. But since they are *all* sparse, the set (taken together) is in fact Dense. For 30 VALUES, 27 Range Cells is quite correct.

This report shows the Histogram for the Metric column USING 200 VALUES

- which produced 113 Cells, 111 of which are Frequency Cells (Single Value Cells)
- all the *actually* repeated Values (Foreign Key lookup) have been identified with Frequency Cells
- the distribution is not even, but that is irrelevant to the intent, which is to get the highly repeated values exposed
- note how the Value Row/Updated has moved, from a Range Cell at number 25 of 25, to a Frequency Cell (Single Value Cell) at 92 of 113.

Table	Column	Cell	F	Value	Pct
SM_ComponentValue	Metric	1		-Null-	
		2	F	APF IO Wait	0.77%
		3		APF Req/Denied IO	0.02%
		4	F	APF Req/Denied Limit	0.30%
		5	F	APF Req/Denied ReUse	0.56%
		6	F	APF Req/Found NoSp	0.84%
		7	F	APF Req/Found Sp	0.59%
		8	F	APF Req/Issued	0.77%
		9	F	APF Requested	0.84%
		10	F	APF Used	0.77%
		11	F	Active Page	1.26%
		12	F	Address Lock	0.40%
		13	F	Buffer Cached	0.84%
		14	F	Buffer Discarded	0.45%
		15	F	Buffer Grabbed	0.84%
		16	F	Buffer Washed	0.84%
		17	F	Buffer Washed/Clean	0.84%
		18	F	Buffer Washed/Dirty	0.84%
		19	F	Byte Received	0.84%
		20	F	Byte Sent	0.64%
		21	F	CPU Busy Pct	0.84%
		22	F	CPU IO Wait	1.26%
		23	F	CPU Idle	1.26%
		88	F	Replacement/Scanned	1.26%
		89	F	Row Affected	0.84%
		90	F	Row/Deleted	0.84%
		91	F	Row/Inserted	0.84%
		92	F	Row/Updated	0.84%
		93	F	Row/Updated/DOL	0.84%
		94	F	Run Queue	1.26%
		95	F	Scope Changed	0.84%
		96	F	Searched	0.84%
		97	F	Statistics Updated	0.84%
		98	F	Swap/Page In	1.26%
		99	F	Swap/Page Out	1.26%
		100	F	System Call	1.26%
		101	F	System Disk Write	0.84%
		102	F	Time Slice Exhausted	0.84%
		103	F	ULC Granted	0.84%
		104	F	ULC Record	0.84%
		105	F	ULC Waited	0.58%
		106	F	Upd/APL Deferred	0.84%
		107	F	Upd/APL Direct	0.84%
		108	F	Upd/DOL Deferred	0.84%
		109	F	Upd/DOL Direct	0.84%
		110	F	User Log Cache	0.58%
		111	F	Voluntary	0.84%
		112	F	Wait Queue	1.26%
		113	F	Xact Committed	0.84%

**Sparse vs Dense**

- With 200 VALUES given to cover 112 distinct Values, Frequency Cells (Single Value Cells) are possible, and have been generated.

**Optimal vs Maximal**

- This shows the *maximal* Cells, the finest grain definition possible for repeated Values for the example, for demonstration purposes
- However, it is not *optimal*:
  - because 2 x 114 Cells are used in sysstatistics for the Histogram, and
  - noting the very small distribution percent for each Value

Eg. 50 Range Cells may be quite qadequate.

