Cheating With Optimiser Statistics In SAP ASE - v2.1

Raymond Mardle

Introduction

- A bit about myself
- How optimiser statistics might be generated
- Tools for analysis
- Customisation procedure
- Other ways of cheating
- Where to find the procedures

Who Is Raymond Mardle?

- I am a relational database specialist
- Apart for two years (2006 and 2007 when I was also using Oracle) I have exclusively used SAP (previously Sybase) products since 1997
  - mainly Adaptive Server Enterprise (ASE)
  - I have various levels of expertise of other SAP products (e.g. Replication Server and IQ)
    - whilst working for Sybase in Australia, I became their Asia / Pacific IQ expert

Who Is Raymond Mardle? (cont)

- I first used Sybase SQL Server 4.9 as a developer in 1989 and then moved into a DBA role (for both Sybase and Oracle) in 1993
- I moved to the Southern Hemisphere in May 1997 to work for ACC in Wellington, New Zealand
- I started working for Sybase Australia in their Melbourne office in August 1998 as a consultant, until the 'great purge' in August 2002
- I moved back to the UK / EU in December 2002

Who Is Raymond Mardle? (cont)

- I am a Certified Sybase Professional and a Certified Sybase Instructor
- I have written several DBA and developer level courses from scratch, and delivered them to employees in-house at two firms I have worked at
- I was the author of the first IQ Quick Reference Guide

Who Is Raymond Mardle? (cont)

- Whilst in a previous employment, I had two articles published in the ISUG Journal
  - Q2 2005 : Surviving Multiple Simultaneous Threshold Firings
  - Q3 2006 : Massaging Statistics in Heterogeneous ASE Environments
    - which was the start point for some of this presentation's content

Simplistic Housekeeping Model

- During a convenient housekeeping window, DBA level jobs run using a single connection to
  - update index statistics for all tables
  - in any time that remains, drop and re-create clustered indexes (or create a dummy CI and then drop it), or use "reorg rebuild" if databases can be dumped afterwards, to defragment and reclaim space (shrink tables)
    - which also rebuilds any non-clustered indexes on the table

Simplistic Housekeeping Model (cont)

- Having as up-to-date as possible statistics is probably more important than having tables as small as they can be – up to the point where the optimiser decides the table is too fragmented and decides to create a new, but usually bad, plan
  - so updating statistics should probably be left to complete, if possible, before performing other housekeeping tasks

Sophisticated Model

- Integrate table shrinkage and stats updating
  - after a table has been shrunk, update the statistics for all indexed columns for the table, except the ones that are first in any index
- Use multiple connections so that
  - after the shrinking, more than one column at a time has its stats updated for the shrunk table
- At the end, update the stats for indexed columns in tables that were not shrunk

Sophisticated Model - Considerations

- There is enough cache available to support more than one column (possibly from different tables) at a time having its stats updating
- There is enough space in the temporary database that the DBA level user uses to allow sorting for several non-leading columns at a time

Shortcomings Of Any Method

- Customised statistics updating commands may be required for certain columns and / or tables, due to
  - different steps being required
  - sampling being needed
- This can be complicated if many servers and / or databases are being administered
- It could take a while for a DBA to react to new requirements
  - during which time the stats may be sub-optimal

Partitioned Tables

- Each partition has a random ID similar to an object ID
- As new partitions are added, the new partition IDs will increase randomly if possible
- If not possible to increase, the new IDs will be lower randomly than existing IDs, so that the IDs are not in partition creation order
- SAP supplied system procedures and utilities output in partition ID order, so they can be out of sequence
- The system procedures discussed in this file can / will output in partition creation order

Tools Available To Analyse Stats

- optdiag (SAP supplied)
- sp_showoptstats (SAP supplied)
- sp__optdiag (originally written by Kevin Sherlock and amended by me)
- sp_rpm_summ_stats (written be me)

optdiag

- SAP supply optdiag
- It is a command line utility that outputs all index information, and stats related information for
  - all tables in a specific database
  - a specific table in a specific database
  - or a specific column in a specific table in a specific database

optdiag (cont)

- The executable is initially found in the ASE software directory / folder structure
- The optdiag version has to match the version of ASE that the executable was created for
- It has to be run by a login with sa_role

optdiag (cont)

- It produces a lot of output
  - which may be far more than is required to, say, find out the last update date and time
- Prior to ASE 16.0, it cannot handle bigtime, bigdate or bigdatetime columns
  - tested in ASE 15.5 EBF 18158 SMP ESD#2 and ASE 15.7 EBF 21338 SMP SP101 on Windows
- It was broken in ASE 16.0 SP03 PL02 on RHEL 7.4 when the default BASH shell language was used

sp_showoptstats

- SAP supply sp_showoptstats
- It was originally "written" using version 15.0 of Kevin Sherlock's sp__optdiag (that is two underscores) as the template
  - none of the bugs in that version were fixed in the "conversion"
  - it took until ASE 16.0 SP03 to fix the bugs
- It only outputs the information in XML

sp_showoptstats (cont)

- As of ASE 16, the XML is built up in a text object
- It usually takes two executions to output the XML
  - the first execution usually fails because the text cannot be output due to textsize being too small
- The XML then has to be run through a parser to allow the statistics to be read
- Has to be run by the table owner or a login with sa_role

sp__optdiag

- Kevin Sherlock wrote the system procedure sp__optdiag
  - its output is similar to optdiag's
- Kevin produced a version for ASE 15.0 and a later version for ASE 16
  - The ASE 15.0 had a few bugs
    - e.g. looping problems for all tables
  - I have not seen Kevin's version for ASE 16.0

sp__optdiag (cont)

- I took Kevin's version for 15.0 and updated it
- I used that to create a version for ASEs 15.5, 15.7 and 16.0
  - the ASE 15.5 and ASE 15.7 versions have not been updated since March 2018

sp__optdiag (cont)

- the ASE 16.0 version was last updated last week
  - it outputs exactly like optdiag by default
  - except for APL round-robin partitioned tables with a CI and one or more NCIs
    - might output additional partition names
- All three versions can output partitions in creation order instead of in partition ID order
  - they can also output extra info that I find to be of use, and can output integer values without decimal places for ASE 16's version

sp__optdiag (cont)

- The procedure does not require sa_role to execute it
- It is created using "with execute as owner" by default, so that statistics functions can be used and statistics can be flushed as part of its processing
- It has the same granularity as optdiag, but with wild cards, and it can have a specific column for multiple tables

sp__optdiag (cont)

- The procedure can be up to about four times faster than optdiag
- For a database with the following table summary (which was created using sp_rpm_tablesize @totals = "J" in a single engine ASE server), where eight of the tables each had 75 columns with statistics

	rowtotal	reserved_KB	data_KB	index_KB	text_im_KB	OAM_KB	unused_KB	cols	inds	stats	parts	allo_u	dis_owner
66 tables. Totals =	1681	20462	1006	1040	448	1612	16356	2873	148	742	477	940	2

Execution time: 0.561 seconds

  - optdiag took 85.25 seconds to create 91,672 lines
  - sp__optdiag took 28:353 seconds to create 91,687 lines

sp__optdiag (cont)

- Supply "?" or "help" as the first parameter for information on its use and output
  - all of the system procedures I write have that functionality

```
sp__optdiag/1.16.0.8/0/B/KJS_n_RPM/AnyPlat/AnyOS/16.0.x/Sun Mar 12 18:30:00 2023

Usage : sp__optdiag [table name
          [, column name
          [, option (not active)
          [, procedure version (not active)
          [, Exactly Like Optdiag (elo) output options
          [, output debug information, with more output for higher values ] ] ] ] ] ]
        where "table name" can use wildcards
              "column name" can use wildcards
              "Exactly Like Optdiag output options" defaults to "y", can also be "Y", "t", "T", "p", "P", "n" or "N"
                if "y" or "Y", the output is exactly like optdiag's output, except with a space on a blank line if "y"
                if "t" or "T", outputs like "y" with the elapsed time at the end
                if "p" or "P", outputs like "y" and sorts any partitions in to creation order, with elapsed time if "P"
                if "n" or "N", outputs like "P" with interger values having no decimal places, and outputs additional information, with uni[var]char converted to strings if "n"

Parameters : sp__optdiag @tabname, @colname, @option, @proc_version, @elo, @debug

Execution time: 0.031 seconds
```

sp_rpm_summ_stats

- Sometimes all that is wanted is to know when the stats for each column in a table were last updated, and possibly some other information
- As part of the investigations for the Q3 2006 ISUG journal article, I wrote (and made available to the Sybase community) a system procedure to summarise statistics information
- I have updated it for use with ASEs 15.0+
- It is now called sp_rpm_summ_stats

sp_rpm_summ_stats (cont)

```
sp_rpm_summ_stats : Version 1.0.5b

Usage : sp_rpm_summ_stats [table name
          [, column name
          [, after datetime
          [, before datetime
          [, minimum rows
          [, only columns that are in an index
          [, only columns that are in a partition
          [, sort column
          [, serial update stats done
          [, output elapsed time in ms ] ] ] ] ] ] ] ] ] ]
        where 'table name' can use wildcards (use 'sys%' for all system tables)
              'column name' can use wildcards
              'after datetime' is the earliest moddate after which to show statistics for
              'before datetime' is the latest moddate before which to show statistics for (useful for finding columns that didn't have their stats updated
                in the most recent maintainance cycle by specifying a value before the most recent cycle started)
              'minimum rows' is the smallest table to show a stats summary for (useful for ignoring empty - use 1 - or very small tables)
              'only columns that are in an index' is Y|y to only have the columns that are in an index, or E|e to exclude them
              'only columns that are in a partition' is Y|y to only have the partitioned stats, E to only have non-partitioned stats with partition counts
                as appropriate, EE to only have non-partitioned stats with partition counts and average values of req_step etc. as appropriate, e to have only
                tables with no partitioned stats, or T|t to have all stats for only partitioned tables (T|t is the opposite of e)
              'sort' is the number of a column in the output to use as the first ordering column, where a positive value does an ascending
                sort or a negative value does a descending sort. The default sort order is ascending on table name, owner, column name and partition condition number.
                (Partition condition number isn't in the output but it is used for sorting, to ensure that the partition stats appear in the correct order.)
                Valid values are -13 to -1 and 1 to 13, or -14 to -1 and 1 to 14 if 'serial update stats done' = Y|y|T|t. A special case of [-]99 will sort
                by [descending] column ID within the table, to give the same column order as when the table was created if positive 99 is specified.
              'serial update stats done' is Y|y if all stats are updated serially, or T|t if more than one table's stats are updated at the
                same time but all the columns for an individual table are done serially (e.g. multiple 'update index stats' running at the same time)
              'output elapsed time in ms' is 1 to do so
```

sp_rpm_summ_stats (cont)

```
Output : If no restrictions are applied, the output will be a summary of all user table columns that are in an index, or that have statistics
          without being in an index

Key : If a table is not partitioned then the ptn column will be blank. If it is partitioned then the ptn column for the summary statistics row will be blank
        and the other rows will contain the partition name. However, if @of_part = 'E' is used to only show non-partitioned stats, i.e. all stats for
        non-partitioned tables and only the summary stats for partitioned tables, then the ptn column will contain a count of the partition stats for a table.
        If @of_part = 'EE' is used then the count will be followed by averages for the req_step, act_step, tune_fac and samp_p columns across the partitioned
        stats of the column.
      The C_summ column shows whether the column is in a clustered index (CI). It can be NULL, 0, 1 or 1.1 - indicating that it isn't in any index
        but that it has statistics, it isn't in a / the CI, it is in the table's CI, or that it is the leading column in the table's CI, respectively.
      The N_summ colum shows how many non-clustered indexes (NCIs) the column is in. It can be NULL, 0, n or n.m - indicating that it isn't in
        any index but that it has statistics, it isn't in any NCIs, it is in 'n' NCIs, or that it is in 'n' NCIs and is the leading column for 'm'
        of them, respectively.
      The Edit column will contain Y if the statistics for the column have been edited in some way - e.g. using sp_modifystats or 'optdiag -i'.
      The moddate column shows when the update stats for a column were done. It is the point where all of the values have been read for the column
        and the processing starts to work out the statistics (e.g. when the sorting starts for non-leading columns of an index). NULL indicates that
        the column is in an index but that it doesn't have any statistics. In that case, the remaining columns will all contain zero.
      The pos_elap column shows how long it may have taken to perform the update stats for a column. It is only shown if @serial_us is Y|y or T|t.
        It is useful for planning maintainance. It is calculated by finding the closest moddate for another column that is less than this column's
        moddate, but with the other column in the same table if T is used. If the column has partitioned statistics, that is also used to find the other moddate.
      An * after the value in the req[uested]_step, tune[ing]_fac[tor] or samp[le]_p[ercent] columns indicates that the value is sticky. The
        act[ual]_step column is never sticky. A 'summary' row for a column in a partitioned table (which is usually the leading column of the index)
        has a tuning factor of zero.

Parameters : sp_rpm_summ_stats @oname, @cname, @dt_after, @dt_before, @min_rows, @in_index, @of_part, @sort, @serial_us, @debug

Execution time: 0.047 seconds
```

sp_rpm_summ_stats (cont)

- It has the same granularity levels as sp__optdiag
- You can also restrict by other criteria
- You can change the sort order of the output
- It handles partitioned tables, and can output partition information in one of several different ways
- If the simple stats updating method is used, it can also output the approximate time each column took to have its stats updated

sp_rpm_summ_stats (cont)

- Example usage : `sp_rpm_summ_stats @serial_us = y`

```
t_name       owner rowcnt  col            ptn  C_summ N_summ Edit moddate             pos_elap req_step  act_step tune_fac samp_p
------       ----- ------  ---            ---  ------ ------ ---- -------             -------- --------  -------- -------- ------
Country      dbo       23 cancelled           0      1           2017.09.27 12:25:00 00:00:00       20         1       20      0
Country      dbo       23 created             0      1.1         2017.09.27 12:25:00 00:00:00       20         2       20      0
Country      dbo       23 id                  1.1    0           2017.09.27 12:25:00 00:00:00       20        20       20      0
Country      dbo       23 name                1      0           2017.09.27 12:25:00 00:00:00       20        20       20      0
Country      dbo       23 r_id                1      0           2017.09.27 12:25:00 00:00:00       20        20       20      0
Country      dbo       23 updated             0      1           2017.09.27 12:25:00 00:00:00       20         1       20      0
Customer     dbo   142182 c_id                0      1.1         2017.09.27 12:25:15 00:00:03       20        46       20      0
Customer     dbo   142182 cancelled           0      1.1         2017.09.27 12:25:22 00:00:04       20        20       20      0
Customer     dbo   142182 created             0      1.1         2017.09.27 12:25:25 00:00:00       20        20       20      0
Customer     dbo   142182 id                  1.1    2           2017.09.27 12:25:25 00:00:00       20        20       20      0
Customer     dbo   142182 name                0      1.1         2017.09.27 12:25:17 00:00:02       20        22       20      0
Customer     dbo   142182 post_code           0      1.1         2017.09.27 12:25:12 00:00:12       20        50       20      0
Customer     dbo   142182 updated             0      1.1         2017.09.27 12:25:23 00:00:01       20        20       20      0
Item         dbo  1458982 created             0      1.1    Y    2017.09.11 15:49:37 NULL           20        20       20      0
Item         dbo  1458982 id                  1.1    0      Y    2017.09.11 15:49:38 00:00:01       20        20       20      0
Item         dbo  1458982 l_id                1      0      Y    2017.09.11 15:50:04 00:00:04       20        25       20      0
Item         dbo  1458982 m_id                1      0      Y    2017.09.11 15:49:56 00:00:05       20        30       20      0
Item         dbo  1458982 name                1      0      Y    2017.09.11 15:49:51 00:00:13       50*      158       20      0
Item         dbo  1458982 s_id                1      0      Y    2017.09.11 15:50:00 00:00:04       20        30       20      0
Item         dbo  1458982 u_id                1      0      Y    2017.09.11 15:50:09 00:00:05       20        12       20      0
Item         dbo  1458982 updated             0      1      Y    2017.09.11 15:50:12 00:00:03       20        20       20      0
Item         dbo  1458982 when_cancelled      0      1      Y    2017.09.11 15:50:15 00:00:03       20        20       20      0
Location     dbo       30 cancelled           0      1           2017.09.27 12:25:46 00:00:00       20         1       20      0
Location     dbo       30 created             0      1.1         2017.09.27 12:25:46 00:00:00       20         2       20      0
Location     dbo       30 id                  1.1    0           2017.09.27 12:25:46 00:00:00       20        20       20      0
Location     dbo       30 position            1      0           2017.09.27 12:25:46 00:00:00       20        20       20      0
Location     dbo       30 updated             0      1           2017.09.27 12:25:46 00:00:00       20         1       20      0
Location     dbo       30 w_id                1      0           2017.09.27 12:25:46 00:00:00       20        20       20      0
                                                                SNIP

(70 rows affected)
(return status = 0)
```

sp_rpm_summ_stats (cont)

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	pos_elap	req_step	act_step	tune_fac	samp_p
Country	dbo	23	cancelled		0	1		2017.09.27 12:25:00	00:00:00	20	1	20	0
Country	dbo	23	created		0	1.1		2017.09.27 12:25:00	00:00:00	20	2	20	0
Country	dbo	23	id		1.1	0		2017.09.27 12:25:00	00:00:00	20	20	20	0
Item	dbo	1458982	m_id		1	0	Y	2017.09.11 15:49:56	00:00:05	20	30	20	0
Item	dbo	1458982	name		1	0	Y	2017.09.11 15:49:51	00:00:13	50*	158	20	0

- Explanation
  - @serial_us = y : outputs the *possible* elapsed time it took to update the column's stats in serial update stats mode with one connection
    - use t if several "update index statistics {table}" were running at the same time using multiple connections, for different tables

sp_rpm_summ_stats (cont)

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	pos_elap	req_step	act_step	tune_fac	samp_p
Country	dbo	23	cancelled		0	1		2017.09.27 12:25:00	00:00:00	20	1	20	0
Country	dbo	23	created		0	1.1		2017.09.27 12:25:00	00:00:00	20	2	20	0
Country	dbo	23	id		1.1	0		2017.09.27 12:25:00	00:00:00	20	20	20	0
Item	dbo	1458982	m_id		1	0	Y	2017.09.11 15:49:56	00:00:05	20	30	20	0
Item	dbo	1458982	name		1	0	Y	2017.09.11 15:49:51	00:00:13	50*	158	20	0

- Explanation (cont)
  - ptn : blank if the table isn't partitioned (stay tuned)
  - C_summ : clustered index summary - can be NULL, 0, 1 or 1.1 : meaning not in any index but has stats, not in a / the CI but in one or more NCIs, in the CI, or is the leading column of the CI, respectively

sp_rpm_summ_stats (cont)

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	pos_elap	req_step	act_step	tune_fac	samp_p
Country	dbo	23	cancelled		0	1		2017.09.27 12:25:00	00:00:00	20	1	20	0
Country	dbo	23	created		0	1.1		2017.09.27 12:25:00	00:00:00	20	2	20	0
Country	dbo	23	id		1.1	0		2017.09.27 12:25:00	00:00:00	20	20	20	0
Item	dbo	1458982	m_id		1	0	Y	2017.09.11 15:49:56	00:00:05	20	30	20	0
Item	dbo	1458982	name		1	0	Y	2017.09.11 15:49:51	00:00:13	50*	158	20	0

- Explanation (cont)
  - N_summ : non-clustered index summary – can be NULL, 0, x or x.y : meaning not in any index but has stats, not in any NCIs but in the CI, in 'x' NCIs, or in 'x' NCIs and is the first column in 'y' of them, respectively
  - a Y in Edit indicates that the stats have been changed in some way after being created

sp_rpm_summ_stats (cont)

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	pos_elap	req_step	act_step	tune_fac	samp_p
Country	dbo	23	cancelled		0	1		2017.09.27 12:25:00	00:00:00	20	1	20	0
Country	dbo	23	created		0	1.1		2017.09.27 12:25:00	00:00:00	20	2	20	0
Country	dbo	23	id		1.1	0		2017.09.27 12:25:00	00:00:00	20	20	20	0
Item	dbo	1458982	m_id		1	0	Y	2017.09.11 15:49:56	00:00:05	20	30	20	0
Item	dbo	1458982	name		1	0	Y	2017.09.11 15:49:51	00:00:13	50*	158	20	0

- Explanation (cont)
  - moddate : this is not the time that the stats were written to sysstatistics
    - it is actually the time that all of the data finished being read before being processed
    - if there is a lot of data that needs sorting, it might be quite a while after this point before the stats are written to sysstatistics

sp_rpm_summ_stats (cont)

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	pos_elap	req_step	act_step	tune_fac	samp_p
Country	dbo	23	cancelled		0	1		2017.09.27 12:25:00	00:00:00	20	1	20	0
Country	dbo	23	created		0	1.1		2017.09.27 12:25:00	00:00:00	20	2	20	0
Country	dbo	23	id		1.1	0		2017.09.27 12:25:00	00:00:00	20	20	20	0
Item	dbo	1458982	m_id		1	0	Y	2017.09.11 15:49:56	00:00:05	20	30	20	0
Item	dbo	1458982	name		1	0	Y	2017.09.11 15:49:51	00:00:13	50*	158	20	0

- Explanation (cont)
  - pos_elap : consequently, this is only *possibly* how long it took to generate the stats for this column using the simple method
    - it is calculated using datediff with this column's moddate and the closest previous moddate
      - use t instead of y if multiple connections did "update index statistics <table>"

sp_rpm_summ_stats (cont)

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	pos_elap	req_step	act_step	tune_fac	samp_p
Country	dbo	23	cancelled		0	1		2017.09.27 12:25:00	00:00:00	20	1	20	0
Country	dbo	23	created		0	1.1		2017.09.27 12:25:00	00:00:00	20	2	20	0
Country	dbo	23	id		1.1	0		2017.09.27 12:25:00	00:00:00	20	20	20	0
Item	dbo	1458982	m_id		1	0	Y	2017.09.11 15:49:56	00:00:05	20	30	20	0
Item	dbo	1458982	name		1	0	Y	2017.09.11 15:49:51	00:00:13	50*	158	20	0

- Explanation (cont)
  - An * after the req_step, tune_fac and / or samp_p value indicates that the value is sticky (but only in ASE 15.7 ESD#2 or greater)

sp_rpm_summ_stats (cont)

- Analysis
  - it is not difficult in this short set of output to spot that the stats for the ***Item*** table were last updated several weeks ago on 11th Sep not 27th Sep (possibly by optdiag, but stay tuned)

```
sp_rpm_summ_stats @serial_us = y

t_name       owner rowcnt  col            ptn  C_summ N_summ Edit moddate             pos_elap req_step  act_step tune_fac samp_p
------       ----- ------  ---            ---  ------ ------ ---- -------             -------- --------  -------- -------- ------
Country      dbo        23 cancelled           0      1           2017.09.27 12:25:00 00:00:00       20         1       20      0
                                                                SNIP
Customer     dbo    142182 updated             0      1.1         2017.09.27 12:25:23 00:00:01       20        20       20      0
Item         dbo   1458982 created             0      1.1    Y    2017.09.11 15:49:37 NULL           20        20       20      0
Item         dbo   1458982 id                  1.1    0      Y    2017.09.11 15:49:38 00:00:01       20        20       20      0
Item         dbo   1458982 l_id                1      0      Y    2017.09.11 15:50:04 00:00:04       20        25       20      0
Item         dbo   1458982 m_id                1      0      Y    2017.09.11 15:49:56 00:00:05       20        30       20      0
Item         dbo   1458982 name                1      0      Y    2017.09.11 15:49:51 00:00:13       50*      158       20      0
Item         dbo   1458982 s_id                1      0      Y    2017.09.11 15:50:00 00:00:04       20        30       20      0
Item         dbo   1458982 u_id                1      0      Y    2017.09.11 15:50:09 00:00:05       20        12       20      0
Item         dbo   1458982 updated             0      1      Y    2017.09.11 15:50:12 00:00:03       20        20       20      0
Item         dbo   1458982 when_cancelled      0      1      Y    2017.09.11 15:50:15 00:00:03       20        20       20      0
Location     dbo        30 cancelled           0      1           2017.09.27 12:25:46 00:00:00       20         1       20      0
                                                                SNIP

(70 rows affected)
(return status = 0)
```

sp_rpm_summ_stats (cont)

- Analysis (cont)
  - to save having to search by eye, such stats can easily be shown by specifying the date of the most recent housekeeping window, as follows

```
sp_rpm_summ_stats @dt_before = "27 Sep 2017", @serial_us = y
```

```
t_name owner rowcnt  col            ptn  C_summ N_summ Edit moddate             pos_elap req_step  act_step tune_fac samp_p
------ ----- ------  ---            ---  ------ ------ ---- -------             -------- --------  -------- -------- ------
Item   dbo   1458982 created             0      1.1    Y    2017.09.11 15:49:37 NULL           20        20       20       0
Item   dbo   1458982 id                  1.1    0      Y    2017.09.11 15:49:38 00:00:01       20        20       20       0
Item   dbo   1458982 l_id                1      0      Y    2017.09.11 15:50:04 00:00:04       20        25       20       0
Item   dbo   1458982 m_id                1      0      Y    2017.09.11 15:49:56 00:00:05       20        30       20       0
Item   dbo   1458982 name                1      0      Y    2017.09.11 15:49:51 00:00:13       50*      158       20       0
Item   dbo   1458982 s_id                1      0      Y    2017.09.11 15:50:00 00:00:04       20        30       20       0
Item   dbo   1458982 u_id                1      0      Y    2017.09.11 15:50:09 00:00:05       20        12       20       0
Item   dbo   1458982 updated             0      1      Y    2017.09.11 15:50:12 00:00:03       20        20       20       0
Item   dbo   1458982 when_cancelled      0      1      Y    2017.09.11 15:50:15 00:00:03       20        20       20       0
(9 rows affected)
(return status = 0)
```

sp_rpm_summ_stats (cont)

- Analysis (cont)
  - what if only 'large tables' are of interest, sorted by stats update date and time with the most recent first? (no pos_elap column this time)

```
sp_rpm_summ_stats @min_rows = 100, @sort = -9
```

```
t_name   owner rowcnt  col            ptn  C_summ N_summ Edit moddate             req_step  act_step tune_fac samp_p
------   ----- ------  ---            ---  ------ ------ ---- -------             --------  -------- -------- ------
Customer dbo    142182 created             0      1.1           2017.09.27 12:25:25       20       20       20      0
Customer dbo    142182 id                  1.1    2             2017.09.27 12:25:25       20       20       20      0
Customer dbo    142182 updated             0      1.1           2017.09.27 12:25:23       20       20       20      0
Customer dbo    142182 cancelled           0      1.1           2017.09.27 12:25:22       20       20       20      0
Customer dbo    142182 name                0      1.1           2017.09.27 12:25:17       20       22       20      0
Customer dbo    142182 c_id                0      1.1           2017.09.27 12:25:15       20       46       20      0
Customer dbo    142182 post_code           0      1.1           2017.09.27 12:25:12       20       50       20      0
Item     dbo   1458982 when_cancelled      0      1      Y      2017.09.11 15:50:15       20       20       20      0
Item     dbo   1458982 updated             0      1      Y      2017.09.11 15:50:12       20       20       20      0
Item     dbo   1458982 u_id                1      0      Y      2017.09.11 15:50:09       20       12       20      0
Item     dbo   1458982 l_id                1      0      Y      2017.09.11 15:50:04       20       25       20      0
Item     dbo   1458982 s_id                1      0      Y      2017.09.11 15:50:00       20       30       20      0
Item     dbo   1458982 m_id                1      0      Y      2017.09.11 15:49:56       20       30       20      0
Item     dbo   1458982 name                1      0      Y      2017.09.11 15:49:51       50*     158       20      0
Item     dbo   1458982 id                  1.1    0      Y      2017.09.11 15:49:38       20       20       20      0
Item     dbo   1458982 created             0      1.1    Y      2017.09.11 15:49:37       20       20       20      0
(16 rows affected)
(return status = 0)
```

sp_rpm_summ_stats (cont)

- Analysis (cont)
  - a partitioned table's ***full summary*** for columns that start with a "v" (look at the rowcnt values)

```
sp_rpm_summ_stats all_types_part, "v%"
```

```
t_name         owner rowcnt col  ptn    C_summ N_summ Edit moddate             req_step act_step tune_fac samp_p
------         ----- ------ ---  ---    ------ ------ ---- -------             -------- -------- -------- ------
all_types_part dbo        4 v1           0      2.1         2017.08.30 11:36:33       20        8       20      0
all_types_part dbo        4 v2           0      1           2017.08.30 11:36:32       20        4        0      0
all_types_part dbo        1 v2   fir_ep  0      1           2017.08.30 11:36:32       20        1       20      0
all_types_part dbo        2 v2   thi_ep  0      1           2017.08.30 11:36:32       20        2       20      0
all_types_part dbo        1 v2   ten_ep  0      1           2017.08.30 11:36:32       20        2       20      0
all_types_part dbo        4 vb1          NULL   NULL        2017.08.30 11:36:33       20        8        0      0
all_types_part dbo        1 vb1  fir_ep  NULL   NULL        2017.08.30 11:36:32       20        2       20      0
all_types_part dbo        2 vb1  thi_ep  NULL   NULL        2017.08.30 11:36:32       20        4       20      0
all_types_part dbo        1 vb1  ten_ep  NULL   NULL        2017.08.30 11:36:33       20        2       20      0
all_types_part dbo        4 vb2          NULL   NULL        2017.08.30 11:36:33       20        4        0      0
all_types_part dbo        1 vb2  fir_ep  NULL   NULL        2017.08.30 11:36:32       20        1       20      0
all_types_part dbo        2 vb2  thi_ep  NULL   NULL        2017.08.30 11:36:32       20        2       20      0
all_types_part dbo        1 vb2  ten_ep  NULL   NULL        2017.08.30 11:36:33       20        2       20      0
(13 rows affected)
(return status = 0)
```

  - ptn contains the name of the partitions with data (the table has 10 partitions in total)

sp_rpm_summ_stats (cont)

- Analysis (cont)
  - the same criteria but with a small summary of a partitioned table's information

```
sp_rpm_summ_stats all_types_part, "v%", @of_part = E
t_name         owner rowcnt col  ptn  C_summ N_summ Edit moddate             req_step act_step tune_fac samp_p
------         ----- ------ ---  ---  ------ ------ ---- -------             -------- -------- -------- ------
all_types_part dbo        4 v1     0  2.1           2017.08.30 11:36:33       20        8       20       0
all_types_part dbo        4 v2     3  0      1      2017.08.30 11:36:32       20        4        0       0
all_types_part dbo        4 vb1    3  NULL   NULL   2017.08.30 11:36:33       20        8        0       0
all_types_part dbo        4 vb2    3  NULL   NULL   2017.08.30 11:36:33       20        4        0       0
(4 rows affected)
(return status = 0)
```

  - rowcnt now contains the total for the table
  - ptn now contains a count of the partitions with data

sp_rpm_summ_stats (cont)

- Analysis (cont)
  - An extended small summary of a partitioned table's summary, sorted by reverse column ID order

```
sp_rpm_summ_stats all_types_part, "v%", @of_part = EE, @serial_us = y, @sort = -99
t_name         owner rowcnt col  ptn         C_summ N_summ Edit moddate              pos_elap req_step act_step tune_fac samp_p
------         ----- ------ ---  ---         ------ ------ ---- -------              -------- -------- -------- -------- ------
all_types_part dbo        4 vb2  3:20/1/20/0 NULL   NULL        2017.08.30 11:36:33 00:00:00       20        4        0      0
all_types_part dbo        4 vb1  3:20/2/20/0 NULL   NULL        2017.08.30 11:36:33 00:00:00       20        8        0      0
all_types_part dbo        4 v2   3:20/1/20/0 0      1           2017.08.30 11:36:32 NULL           20        4        0      0
all_types_part dbo        4 v1               0      2.1         2017.08.30 11:36:33 00:00:00       20        8       20      0
(4 rows affected)
(return status = 0)
```

    - ptn now contains a count of the partitions with data, and the averages of the req_step, act_step, tune_fac and samp_p values for those partitions

Progress

- A bit about myself ✓
- How statistics might be generated ✓
- Tools for analysis ✓
- Customisation procedure
- Other ways of cheating
- Where to find the procedures

Cheating With Statistics 1

- It is unlikely that every column in every table in every database will need (or want) to have the same number of steps, tuning factor or sampling
- Customised stats updating jobs could be written to handle the different requirements for such columns
- That could be a lot of extra work to set up and to maintain properly

Cheating With Statistics 1 (cont)

- Some tables may require some or all columns to have their stats updated more often than during the regular housekeeping window
  - which could mean different scripts for different days
- When a schema change occurs, stats updating before any testing can be done could add a considerable amount of time to the process

Cheating With Statistics 1 (cont)

- I've written four procedures to assist with the issues mentioned
  - sp_rpm_custom_stats (made available as part of the Q3 2006 ISUG article but updated for ASEs 15.5+)
  - sp_rpm_copy_stats
  - sp_rpm_shuffle_stats
  - sp_rpm_append_stats

sp_rpm_custom_stats

- It's first incarnation was written for ASE 12.0
- It allowed the requested steps for a column to be changed from
  - the server's default
  - a value specified as part of a previous update stats command
  - or the value inherited when stats were held on a single page in earlier versions
- Once changed, using "update [index] statistics" would use that new requested step value without need for further customisation (i.e. it was sticky)

sp_rpm_custom_stats (cont)

- I built-in the ability to copy requested step settings from a configuration table, or from an existing table
  - useful for schema changes made as follows
    - rename the existing table
    - create a new version of the table
    - populate the new version of the table (possibly done at the same time as the creation)
    - create the indexes
    - update the statistics
    - drop the renamed table if everything OK

sp_rpm_custom_stats (cont)

- use the procedure at any point after creating the new version of the table and before creating the first index

```
sp_rename Item, Item_save
Object name has been changed.
Warning: Changing an object or column name could break existing stored procedures, cached statements or other
compiled objects.
(1 row affected)
(return status = 0)
select *, who_cancelled = convert (varchar (255), null) into Item from Item_save where 1 = 2
(0 rows affected)
sp_rpm_custom_stats Item, @action = "All", @sourcetable = Item_save
Column name (ID = 2) of table Item (ID = 937051343) in database T5 (ID = 10) does not have a formatid 100 row in
sysstatistics
Two rows for requested steps for column name (ID = 2) of table Item (ID = 937051343) inserted into sysstatistics
for database T5 (ID = 10), with value 50
(return status = 0)
sp_rpm_summ_stats Item
t_name owner rowcnt col  ptn  C_summ N_summ Edit moddate             req_step  act_step tune_fac samp_p
------ ----- ------ ---  ---  ------ ------ ---- -------             --------  -------- -------- ------
Item   dbo        0 name      NULL   NULL        2017.09.27 18:26:33       50*         1       20      0
(1 row affected)
(return status = 0)
```

sp_rpm_custom_stats (cont)

sp__optdiag Item, name, @elo = n

```
sp__optdiag/1.16.0.5/0/B/KJS_n_RPM/AnyPlat/AnyOS/16.0.x/Thu Sep 14 16:07:00 2017
Adaptive Server Enterprise/16.0 SP02 PL02/EBF 25319 SMP/P/X64/Windows Server/ase160sp02plx/0/64-bit/FBO/Sun Nov 22 05:16:54 2015
                                                                SNIP
Statistics for column:                                                  "name"
Column Number:                                                          2
Last update of column statistics:                                       Sep 27 2017  6:26:33:773PM

     Range cell density:                                                0.0000000000000000
     Total density:                                                     1.0000000000000000
     Range selectivity:                                                 default used (0.33)
     In between selectivity:                                            default used (0.25)
     Unique range values:                                               default used (0.000000)
     Unique total values:                                               default used (1.000000)
     Average column width:                                              default used (255.00)
     Rows scanned:                                                      default used (null)
     Statistics version:                                                0

Histogram for column:                                                   "name"
Column datatype:                                                        varchar(255)
Requested step count:                                                   50
Actual step count:                                                      1
Sampling Percent:                                                       0
Tuning Factor:                                                          20
Out of range Histogram Adjustment is DEFAULT.
Sticky step count.

     Step     Weight                    Value

        1     1.00000000      =     null

No statistics for remaining columns:                                    "created"
(default values used)                                                   "description"
                                                                        "id"
                                                                SNIP
                                                                        "when_cancelled"
                                                                        "who_cancelled"


Elapsed = 00:00:00:033
sp__optdiag succeeded.
```

sp_rpm_custom_stats (cont)

- The procedure calls itself when copying settings, once for each value that needs to be copied

```
sp_rpm_custom_stats Item, @action = "All", @sourcetable = Item_save
Column name (ID = 2) of table Item (ID = 969051457) in database T5 (ID = 10) does not have a formatid 100 row in
sysstatistics
Two rows for requested steps for column name (ID = 2) of table Item (ID = 969051457) inserted into sysstatistics for
database T5 (ID = 10), with value 50
Column when_cancelled (ID = 15) of table Item (ID = 969051457) in database T5 (ID = 10) does not have a formatid 100 row
in sysstatistics
Two rows for tuning factor for column when_cancelled (ID = 15) of table Item (ID = 969051457) inserted into
sysstatistics for database T5 (ID = 10), with value 30
Column id (ID = 1) of table Item (ID = 969051457) in database T5 (ID = 10) does not have a formatid 100 row in
sysstatistics
Two rows for sampling percentage for column id (ID = 1) of table Item (ID = 969051457) inserted into sysstatistics for
database T5 (ID = 10), with value 50
(return status = 0)
```

```
sp_rpm_summ_stats Item
```

t_name	owner	rowcnt	col	ptn	C_summ	N_summ	Edit	moddate	req_step	act_step	tune_fac	samp_p
Item	dbo	0	id		NULL	NULL		2017.09.27 18:43:13	20	1	20	50*
Item	dbo	0	name		NULL	NULL		2017.09.27 18:43:12	50*	1	20	0
Item	dbo	0	when_cancelled		NULL	NULL		2017.09.27 18:43:13	20	1	30*	0

```
(3 rows affected)
(return status = 0)
```

sp_rpm_custom_stats (cont)

- The single system procedure execution above is equivalent to the following three executions, but no knowledge of the current settings is required

```
sp_rpm_custom_stats Item, id, NULL, Sampling, "50"
Column id (ID = 1) of table Item (ID = 1001051571) in database T5 (ID = 10) does not have a formatid 100 row in sysstatistics
Two rows for sampling percentage for column id (ID = 1) of table Item (ID = 1001051571) inserted into sysstatistics for database T5 (ID =
10), with value 50
(return status = 0)
sp_rpm_custom_stats Item, name, NULL, ReqStep, "50"
Column name (ID = 2) of table Item (ID = 1001051571) in database T5 (ID = 10) does not have a formatid 100 row in sysstatistics
Two rows for requested steps for column name (ID = 2) of table Item (ID = 1001051571) inserted into sysstatistics for database T5 (ID =
10), with value 50
(return status = 0)
sp_rpm_custom_stats Item, when_cancelled, NULL, TuneFac, "30"
Column when_cancelled (ID = 15) of table Item (ID = 1001051571) in database T5 (ID = 10) does not have a formatid 100 row in sysstatistics
Two rows for tuning factor for column when_cancelled (ID = 15) of table Item (ID = 1001051571) inserted into sysstatistics for database T5
(ID = 10), with value 30
(return status = 0)
sp_rpm_summ_stats Item
t_name owner rowcnt col           ptn  C_summ N_summ Edit moddate             req_step  act_step tune_fac  samp_p
------ ----- ------ ---           ---  ------ ------ ---- -------             --------  -------- --------  ------
Item   dbo        0 id                 NULL   NULL       2017.09.27 18:48:04       20         1       20      50*
Item   dbo        0 name               NULL   NULL       2017.09.27 18:48:39       50*        1       20       0
Item   dbo        0 when_cancelled     NULL   NULL       2017.09.27 18:48:59       20         1       30*      0
(3 rows affected)
(return status = 0)
```

sp_rpm_custom_stats (cont)

- As well as the settings seen above, it can also be used to change Hashing, RangeAbsolute, RangeFactor, TotalAbsolute or TotalFactor
- ReqStep, TuneFac, Sampling and Hashing make changes directly to sysstatistics

sp_rpm_custom_stats (cont)

- RangeAbsolute, RangeFactor, TotalAbsolute and TotalFactor use sp_modifystats to make changes and there must be existing statistics to change
  - changing any of them marks the column as having edited statistics
- Only ASE 15.7 ESD#2+ allows *all* of the settings above to be changed
  - earlier versions cannot change TuneFac, Sampling or Hashing

sp_rpm_custom_stats (cont)

- The shortcoming of this system procedure is that statistics have to be updated after the settings have been customised
  - but not after changing RangeAbsolute, RangeFactor, TotalAbsolute or TotalFactor
- However, it allows for a single stats updating script to be used, with no customisation needed for different tables and columns
  - if columns do need different values, using this procedure is likely easier than changing scripts

sp_rpm_custom_stats (cont)

```
sp_rpm_custom_stats : Version 2.2.0

Usage : sp_rpm_custom_stats destination_table_name [, column_name [, partition_name [, action
            [, 'value_1' [, 'value_2' [, source_table_name [, silent [, debug ]]]]]]]]

Where : destination_table_name is the name of the table to have its statistics attributes changed (which can be prefixed with the owner name)
        column_name is the name of the column to have its statistics attributes changed, or NULL
        partition_name is the name of a partition, NULL, or AllParts to apply the required change to all partitions
        action is one of ReqStep, TuneFac, Sampling, Hashing, RangeAbsolute, RangeFactor, TotalAbsolute, TotalFactor or All
        'value_1' is the value to change the statistics attribute to for the first four actions - specify in quotes if a number
        'value_2' is used by the four Range and Total actions - specify in quotes if a number
        source_table_name is the name of a data table to take statistic attribute information from, or the configuration table name
          (which must be CONFIQ_COLUMN_STAT) that holds information about what statistic attributes are to be changed for one or more tables
          (which can be prefixed with the owner name)
        silent is a flag that causes information messages to be supressed if set to Y|y
        debug outputs tracing information to assist in fault finding

        ReqStep is used to change requested steps for a column, and make it sticky in ASE 15.7+
        TuneFac is used to change the tuning factor for a column and make it sticky (only in ASE 15.7+)
        Sampling is used to change the sampling for a column and make it sticky (only in ASE 15.7+)
        Hashing is used to change the hashing for a column and make it sticky (only in ASE 15.7+)
        RangeAbsolute, RangeFactor, TotalAbsolute or TotalFactor control the density setting using sp_modifystats
        An action of All is used to change appropriate settings above when a source table is used. In pre-ASE 15.7, it will only change ReqStep as appropriate if
          CONFIQ_COLUMN_STAT isn't used. In ASE 15.7+, it will change ReqStep, TuneFac, Sampling and Hashing as appropriate if CONFIQ_COLUMN_STAT isn't used.
        Using config as 'value_1' will set the required statistics attribute to the corresponding configuration setting's value. It will also clear
          any stickyness for the action when ReqStep, TuneFac and Sampling; and stickyness for Hashing if the configuration setting is 'off'.

Parameters : @destable, @column, @partn, @action, @val1, @val2, @sourcetable, @silent = 'N', @debug = 0

Elapsed = 00:00:00:016
Execution time: 0.047 seconds
```

sp_rpm_copy_stats

- This system procedure looks like it makes changes to stats settings just like sp_rpm_custom_stats
- However, it copies the statistics themselves to be for the new version of the table

sp_rpm_copy_stats (cont)

- In the schema change scenario described above, the new version of the table is populated from the existing version of the table
- If the only new (or changed) data is in new columns, then the existing statistics are still valid for the new version of the table
- Consequently, the statistics on the existing data are still valid in the new version of the table

sp_rpm_copy_stats (cont)

- I've written this system procedure so that the new version of the table does not need to have the same layout as the existing version
  - columns can move position
  - columns can change names and / or datatype
  - partitions can change names
  - functional indexes can change position in the index creation order

sp_rpm_copy_stats (cont)

- Statistics would only have to be updated for columns that are now in an index which weren't previously in an index
- If there are no functional indexes, this procedure can be executed before any indexes are created
  - otherwise it has to be executed after the last functional index is created
- Indexes can be created specifying "with 0 values", which may be an additional saving in time

sp_rpm_copy_stats (cont)

- See the two examples of using sp_rpm_copy_stats on the web page (it's link will be given at the end of the presentation) for the full proof that this process works
- The next slide has the existing and new versions of the table used in the second example of the proof
  - apologies for the small font and how much is on the next slide

sp_rpm_copy_stats (cont)

```
create table Item
(id numeric (6) NOT NULL,
 name varchar (30) NOT NULL,
 m_id char (3) NOT NULL,
 s_id char (3) NOT NULL,
 l_id char (3) NOT NULL,
 u_id char (3) NOT NULL,
 quantity smallint NOT NULL,
 description varchar (50) NOT NULL,
 price smallmoney NOT NULL,

 stock smallint NOT NULL,

 created smalldatetime NOT NULL,
 updated smalldatetime NULL,
 cancelled smalldatetime NULL) lock allpages partition by range (id)
 (p01 values <= (109999),
  p02 values <= (119999),
  p03 values <= (129999),
  p04 values <= (139999),
  p05 values <= (149999),
  p06 values <= (159999),
  p07 values <= (169999),
  p08 values <= (179999),
  p09 values <= (189999),
  p10 values <= (199999),
  pma values <= (MAX))

/* Populated */

create clustered index Item_ci on Item (id, m_id, s_id, l_id, u_id, name)
 with statistics using 0 values

create index Item_nci_1 on Item (created, updated, cancelled)
 with statistics using 0 values



create index Item_fc_nci_1 on Item (price * stock)
 with statistics using 0 values local index fc_Part

exec sp_rpm_custom_stats Item, id, p01, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p02, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p03, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p04, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p05, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p06, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p08, ReqStep, "30"
exec sp_rpm_custom_stats Item, id, p09, ReqStep, "30"

update index statistics Item

delete statistics Item (created)

update statistics Item (created) with print_progress = 1
```

```
create table Item
(id bigint NOT NULL,
 name varchar (255) NOT NULL,
 m_id varchar (10) NOT NULL,
 s_id varchar (10) NOT NULL,
 l_id varchar (10) NOT NULL,
 u_id varchar (10) NOT NULL,
 unit_quantity int NOT NULL,
 description varchar (255) NOT NULL,
 price money NOT NULL,
 reserved int NULL,
 in_stock int NOT NULL,
 on_order int NULL,
 created datetime NOT NULL,
 updated datetime NULL,
 when_cancelled smalldatetime NULL) lock allpages partition by range (id)
 (p01 values <= (109999),
  p02 values <= (119999),
  p03 values <= (129999),
  p04 values <= (139999),
  p05 values <= (149999),
  p06 values <= (159999),
  p07 values <= (169999),
  p08 values <= (179999),
  p09 values <= (189999),
  p10 values <= (199999),
  pmax values <= (MAX))

/* Insert */

create clustered index Item_ci on Item (id, m_id, s_id, l_id, u_id, name)
 with statistics using 0 values

create index Item_nci_1 on Item (created, updated, when_cancelled)
 with statistics using 0 values

create index Item_nci_2 on Item (u_id, l_id, s_id, m_id, updated)
 with statistics using 0 values

create index Item_fc_nci_1 on Item (price * in_stock)
 with statistics using 0 values local index fc_Part
```

sp_rpm_copy_stats (cont)

- Creating the functional index on the populated new version of the table re-creates the table, which rebuilds the indexes, *which creates stats for the leading column of each of the indexes, even though "using 0 values" is specified* **(a bug, methinks)**

sp_rpm_copy_stats (cont)

- Afterwards , the statistics can be copied :
  @del_existing = Yes is used because of the above

```
sp_rpm_copy_stats Item_save, Item, @col_manual = "quantity = unit_quantity,
     sybfi3_1 = sybfi4_1", @part_manual = "pma = pmax", @del_existing = Yes, @debug = 1
  --  @col_manual does not need 'quantity = unit_quantity' because they don't have
statistics but it does need 'sybfi3_1 = sybfi4_1'; cancelled / when_cancelled was
matched on datatype because they were the only unmatched columns with the same datatype
(smalldatetime); the name of the maximum partition was changed, so that needs to be
specified in @part_manual

Modified the first formatid 102 cell of source table Item_save's 'created' column to be
the end of the minute / day when converting from smalldatetime to datetime for 10
partitions
Modified the first formatid 102 cell of source table Item_save's 'updated' column to be
the end of the minute / day when converting from smalldatetime to datetime for 10
partitions
Updated 194 source table temporary statistics formatid 102 and 100 rows for columns
changing datatypes between the two tables
Deleted 96 rows from and inserted 392 rows into sysstatistics for copying statistics
for source table Item_save and destination table Item
Elapsed = 00:00:01:606
(return status = 0)
```

sp_rpm_copy_stats (cont)

- Doing "update index statistics Item" for the new version of the table, with its database devices in RAMDisk and an in-memory tempdb, took 00:00:32:980 (and much longer when the devices were on HDD based devices)
- To get partitioned stats for the leading column of the two NCIs (created and u_id), for better optimiser processing, took a further 00:00:10:780
- Compared to 00:00:01:606 to copy them using sp_rpm_copy_stats

sp_rpm_copy_stats (cont)

- The two proofs on the web page show that ASE doesn't care how the stats for a table got into sysstatistics
  - it will use what it has available for any new plan creations
- So although this has a limited scope of use, it could be very useful if used as part of the schema change process

sp_rpm_copy_stats (cont)

- sp_rpm_copy_stats is not a replacement for updating stats for new versions of a table
- It is to allow testing of schema changes to start sooner than might otherwise be the case if one or more very large tables are being changed

sp_rpm_copy_stats (cont)

```
sp_rpm_copy_stats : Version 2.0.3

Usage : sp_rpm_copy_stats source_table_name, destination_table_name [, manual_column_information
                          [, manual_index_information [, manual_partition_information, [,delete_existing [, silent
                          [, transient table owner override [, source table owner ID override [, source table ID override
                          [, debug ]]]]]]]]

Where : source_table_name is the name of the table to copy the statistics information for
        destination_table_name is the name of the table that will end up with the copied statistics information
        manual_column_information is a list of column names that have been renamed, in the form '<original name>=<new name>[ |,|;][...]'
        manual_index_information is a list of index IDs or table types, for indexes that have been renamed or have moved index ID,
          in the form '[+{1|2|3}|<original ID>=<new ID>[ |,|;][...]' or '{apl|dol}={dol|apl}[ |,|;][...] : use '+1' or 'apl=dol' when the table has
          changed from allpages to datapages or datarows, or 'dol=apl' for the converse : it is added in anticipation of index IDs being stored in sysstatistics
        manual_partition_information is a list of partition names that have been renamed, in the form '<original name>=<new name>[ |,|;][...]'
        delete_existing has to be 'Yes' to force the deletion of existing destination table statistics that the source table has available for copying
        silent is a flag that causes information messages to be suppressed when set to Y|y
        transient table owner override is the username in tempdb of the user that owns the transient system data tables in tempdb to contain the data for copying statistics
        source table owner ID override is the ID of the owner of the source table that statistics are being copied for
        source table ID override is the ID of the source table that statistics are being copied for
        debug outputs tracing information to assist in fault finding
```

sp_rpm_copy_stats (cont)

```
Description : The statistics of the source table are manipulated as required to make them valid for the layout of the new version of the source table,
                which is the destination table. It automatically cross matches any column, index (when required) and partition (if used) names that are the same
                in both tables. For list and range partitioned tables, the conditions will be used for those not matched on name. Any partitions that do not match
                on name or condition will be matched on position.
              A column not matched by name will automatically be matched by datatype if there is only one unmatched column in the two tables with the same datatype.
              Columns, indexes and partitions can be manually cross matched using the appropriate @manual parameter.
              If any existing columns that have statistics have changed position in the destination table, then the stats are modified to reflect that
                by modifying the colidarray value(s).
              If any existing columns that have statistics have changed datatype in the destination table, then the stats are modified to reflect that,
                as long as it is possible to do an explicit conversion between the two datatypes : e.g. varchar to integer will only work if the string
                only contains digits, zero or one minus signs, and zero or one dots.
              If the destination table already has statistics that have the same identifiers as the results of the manipulation of the source table's statistics,
                then they will remain by default. To use all of the source table's statistics, specify '@del_existing = Yes' to remove the destination table's
                statistics that are waiting to be copied.
              The statistics can be sourced from transient tables loaded into tempdb, allowing the statistics to be copied between ASE servers.

Information : If the destination table ends up with statistics for a binary column of any length, it is likely that trailing zeros will be lost after the processing
                is complete. I am unable to find a way to restore the trailing zeros but they do not appear to impact the actual value stored prior to the lost zeros
                when tested in Adaptive Server Enterprise/16.0 SP02 PL02/EBF 25319 SMP/P/X64/Windows Server/ase160sp02plx/0/64-bit/FBO/Sun Nov 22 05:16:54 2015

Parameters : @sourcetable, @desttable, @col_manual, @ind_manual, @part_manual, @del_existing = 'NO', @silent = 'N',  @st_use_over, @st_uid_over, @st_id_over, @debug = 0

Elapsed = 00:00:00:000
Execution time: 1.591 seconds
```

Cheating With Statistics 2

- My manager in a previous employment asked for many years for Sybase / SAP to supply a mechanism for what I am about to describe
- They never did so
- This is one example of someone else saying "wouldn't it be great if ..." and me devising and creating a solution
- When I had a long period to work on my own projects, I made this one of the issues I tackled

sp_rpm_shuffle_stats

- This was written for a situation that most people are never likely to encounter
  - a live table is one that contains data for the current date, and it has around the same number of rows for every date
  - a history table has the same layout as its equivalent live table, but with one extra column that contains the date that the live data relates to
  - an archive table has the same layout as the equivalent history table

sp_rpm_shuffle_stats (cont)

- at the end of the business day, the live data is copied to the relevant history table
- then data older than 'x' days is
  - copied from the history table to relevant archive table (if there is one)
  - deleted from the history table
- then data older than 'y' days is
  - deleted from the archive table (if there is one)

sp_rpm_shuffle_stats (cont)

- The sizes of the live, and the history and archive (if there is one) tables stay at around the same levels every day, once the non-live(s) days are fully populated
- After the copying and deleting is complete for a set of tables, the history and archive tables have around the same number of rows as they did before the processing started but there are
  - statistics for a date that is no longer in the tables
  - no statistics for the newest date

sp_rpm_shuffle_stats (cont)

- A live table can contain many thousands of rows
  - maybe 10's or 100's of thousands of rows
- The history and archive tables will contain 'x' times and ('y' – 'x') times many thousands of rows, respectively
  - with several columns in multiple indexes, stats updating for these tables is time consuming
    - even if just the date column has its stats updated

sp_rpm_shuffle_stats (cont)

- Having tables like these allows live data to be accessed and changed throughout the day in a moderately sized table
  - that can have its statistics updated every day if required
- Reporting using older data can occur without impacting the accessibility of the live data

sp_rpm_shuffle_stats (cont)

- All of the above live, history and archive table usage was devised and implemented before tables could be partitioned
- There are no plans to partition the history and archive tables

sp_rpm_shuffle_stats (cont)

- Statistics are only updated on the history and archive tables once a week at the most
  - other work (e.g. schema changes) during the housekeeping window may impact on how much time is available for updating statistics
  - consequently, some history and / or archive tables may not have their statistics updated for several weeks

sp_rpm_shuffle_stats (cont)

- As of ASE 15.7 ESD#2 onwards, 'out of range' can be set on the date column
- However, as more dates are added, the extrapolation that that allows becomes less accurate
- sp_rpm_shuffle_stats was written to try and have more accurate statistics throughout the working week

sp_rpm_shuffle_stats (cont)

- It requires that each date has sparse frequency cells, with two entries per date (stay tuned)
  - the first entry has < the date and a weight of zero
  - the second has = the date and the weight for the date
  - all dates have a similar non-zero weight

sp_rpm_shuffle_stats (cont)

- There must be no rows in the table for the pair of cells with the oldest date
  - which are steps 1 and 2 in the histogram output
- There must be around the same number of rows for the newly added date as there were for the oldest date that were deleted
  - i.e. the new date's weight must be very similar to the oldest date's weight

sp_rpm_shuffle_stats (cont)

- The weight for the oldest date is saved (which is the value in c1 of the first formatid 104 row for the date column)
- The pairs of cells are moved down one set for the formatid 102 and 104 rows
  - e.g. c2 → c0 & c3 → c1, c4 → c2 & c5 → c3 . . .
  - if there is more than one formatid 104 row (i.e. more than 40 dates), c0 and c1 of the next row become c78 and c79 of this row

sp_rpm_shuffle_stats (cont)

- The pair of cells with the previous oldest date get changed to have
  - the current date's value (the new maximum value)
  - the weight saved in the first point above
- By cycling the first weight to be the last weight, the total for the weights remains the same as it was before
  - which should be exactly one, or very close to it

sp_rpm_shuffle_stats (cont)

- This only works for [small|big]datetime columns
- If a date column is used
  - a set of dense frequency cells are generated if all of the dates are contiguous
  - a combination of dense and sparse frequency cells are generated if there are any gaps in the date sequence
- It is unlikely that I'll extend the functionality to be able to cope with the above

sp_rpm_shuffle_stats (cont)

- The table must be fully populated with all of the requisite number of dates of data
  - stay tuned to find out how to handle a table used for holding a finite number of dates not yet being fully populated for all dates, by using a different way of cheating
- There can only be one new date's worth of data to process per execution

sp_rpm_shuffle_stats (cont)

- The procedure does not work against partitioned tables
  - a set of partitioned statistics contains a summary set and a set for each partition with data
  - shuffling none, some or all of the sets of statistics might break something, so I decided to leave them alone for this (and the next) procedure

sp_rpm_shuffle_stats (cont)

```
sp_rpm_shuffle_stats : Version 2.0.0

Usage : sp_rpm_shuffle_stats table_name, column_name [, maximum_weight_difference_percentage [, silent [, debug ]]]

Where : table_name is the name of the table to have its statistics values and weights shuffled down (which can be prefixed with the owner name)
        column_name is the name of the column to have its statistics values and weights shuffled down - ideally a [small|big]datetime column
        maximum_weight_difference_percentage is the largest percentage difference that can exist between two contiguous pairs of weights
        silent is a flag that causes information messages to be suppressed when set to Y|y
        debug outputs tracing information to assist in fault finding

Description : If each unique grouping value in the column has a pair of stats entries (with the first of the pair having a value of NULL and a weight of 0, and the
                second of the pair having the unique grouping value and a non-zero weight), the grouping value for the first pair does not have any data, and
                there is only one new grouping value that is greater than the current maximum grouping value in the stats; then the first pair will be overwritten by
                the second pair, the second pair will be overwritten by the third pair, etc., and the last pair will be overwritten by the new highest grouping
                value and the weights from the first pair.
              This is only valid if the weights in each pair are approximately the same - i.e. there are approximately the same large number of rows for each unique
                grouping value.
              Due to the way that partitioned tables have statistics stored, with stats for each partition and summary stats for all partitions, the stats for
                partitioned tables cannot be shuffled. This ability is not quite as necessary for them, as a partition's stats will probably be quicker to update.

Parameters : @shuftable, @column, @max_diff_pct = 1.1, @silent = 'N', @debug = 0

Elapsed = 00:00:00:000
Execution time: 0.67 seconds
```

sp_rpm_append_stats

- If a history or archive table is not fully populated, until the stats are next updated, then each new date's worth of data will
  - degrade the "out of range" extrapolation if that is set
  - cause larger and larger errors for the number of rows estimated for dates with existing stats
    - as the optimiser applies the weight to the row count

sp_rpm_append_stats (cont)

- sp_rpm_append_stats appends two sparse frequency cells for the next date with data after the set with the most recent date in the statistics
  - it also massages all of the existing weights, and other information about the column
    - e.g. total density

sp_rpm_append_stats (cont)

- If each set of data had exactly the same number of rows, then each weight would be
  - 1 / 'number of dates'
- However, there will usually be a small percentage difference in the number of rows for each date
  - so the existing weights and the new weight have to be massaged based upon this small percentage difference

sp_rpm_append_stats (cont)

- The current average weight (caw) is (1 / 'number of dates')
- The new average weight (naw) is (1 / ('number of dates' + 1))
- The weight difference factor (wdf) is (caw / naw)
- Each existing weight is massaged using
  - ((weight – caw) / wdf) + naw
- Testing has shown that the resulting massaged value is 'close' to what a new stats update generates after the data for the new date is added

sp_rpm_append_stats (cont)

- To calculate the weight for the new pair
  - a running total of ((weight – caw) / wdf) is calculated
  - the new weight is (naw – final running total)
- The table can contain multiple sets of new dates' data
  - each execution of sp_rpm_append_stats will only process the next date after the most recent date with statistics

sp_rpm_append_stats (cont)

- In a table with stats for 31 dates' data with around 510 rows per set of data
  - a new date's data were inserted, with 516 rows
  - sp_rpm_append_stats was executed

```
sp_rpm_append_stats B_HIST, dt, @max_diff_pct = 3.3, @debug = 1
Appended new stats value 'Mar  5 2017 12:00:00.000000AM' with weights
0x00000000 and 0.0312499646 for column dt in the table B_HIST
Each stored procedure and trigger that uses table 'B_HIST' will be
recompiled the next time it is executed.
Elapsed = 00:00:00:080
(return status = 0)
```

sp_rpm_append_stats (cont)

- A higher maximum difference percentage of 3.3 than the default of 1.1 had to be specified due to the low number of rows for each date
- the percentage difference ensures that the existing weights are not too different from one to the next
- the 32 dates had between 501 and 519 rows
- the largest row count difference between two contiguous dates was 11

sp_rpm_append_stats (cont)

- the massaged weight for the new date's pair was 0.03124996
- its weight after stats updating the 32 dates' data was 0.03147109, a -0.708% difference

sp_rpm_append_stats (cont)

- In a table with stats for 31 dates' data with around 510 rows per set of data
  - 24 new dates' of data were inserted, with around 510 rows each
  - append_stats was executed 24 times
  - the massaged weight for the 24th new date's pair was 0.01818181
  - its weight after stats updating the 55 dates' data was 0.01830371, a -0.670% difference

sp_rpm_append_stats (cont)

- In a situation where a new set of live, history and possibly archive tables are needed, or the number of dates' of data needs to be increased in the history and / or archive tables
  - and it isn't possible to update the stats for each new date's data after being inserted
  - then sp_rpm_append_stats will allow the optimiser to make better estimates between stats updating

sp_rpm_append_stats (cont)

```
sp_rpm_append_stats : Version 1.0.0

Usage : sp_rpm_append_stats table_name, column_name [, maximum_weight_difference_percentage [, silent [, debug ]]]

Where : table_name is the name of the table to have its statistics values and weights appended to (which can be prefixed with the owner name)
        column_name is the name of the column to have its statistics values and weights appended to - ideally a [small|big]datetime column
        maximum_weight_difference_percentage is the largest percentage difference that can exist between two contiguous pairs of weights
        silent is a flag that causes information messages to be suppressed when set to Y|y
        debug outputs tracing information to assist in fault finding

Description : If each unique grouping value in the column has a pair of stats entries (with the first of the pair having a value of NULL and a weight of 0, and the
                second of the pair having the unique grouping value and a non-zero weight), the grouping value for the first pair has data, and there is one or more
                new grouping value that is greater than the current maximum grouping value in the stats; then a new set of stats will be appended for the first
                grouping value greater than the current most recent grouping value with stats, with the weight calculated from the current and new average weights.
              This is only valid if the weights in each pair are approximately the same - i.e. there are approximately the same large number of rows for each unique
                grouping value.
              The weights for the existing stats and the summary stats row for the column will be massaged to make them better reflect the new number of grouping values.
              Due to the way that partitioned tables have statistics stored, with stats for each partition and summary stats for all partitions, the stats for
                partitioned tables cannot be appended to. This ability is not quite as necessary for them, as a partition's stats will probably be quicker to update.

Parameters : @apptable, @column, @max_diff_pct = 1.1, @silent = 'N', @debug = 0

Elapsed = 00:00:00:000
Execution time: 0.736 seconds
```

Permissions

- The four cheating statistics system procedures can be executed by
  - a user with sa_role
  - the owner of the database that the table is in
  - the owner of the table
  - a user with "update statistics" permission
    - permission granted either directly or via a role (down to four levels of role nesting)

Where To Find The Procedures

- I have a web page hanging off of the side of the web site I administer for the Lumphanan Community Recreation Association (LCRA)
  - we host the first 10 KM run of the year in Scotland, held on the 2nd of January every year (weather and pandemics permitting)
  - it is called "The Lumphanan Detox 10K"
  - please get in touch if your firm would like to help sponsor a race

`https://www.lumphanan.com/ase`

Summary

- A bit about myself ✓
- How statistics might be generated ✓
- Tools for analysis ✓
- Customisation procedure ✓
- Other ways of cheating ✓
- Where to find the procedures ✓