Cheating With Statistics In SAP ASE

Raymond Mardle

Introduction

- A bit about myself
- How 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, 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, 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
- I was the author of the first IQ Quick Reference Guide

Who Is Raymond Mardle? (cont)

- Whilst working for JP Morgan Chase, 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 the 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 the non-clustered indexes

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
  - 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
  - more than one table is shrunk at a time

Sophisticated Model - Considerations

- There is enough free space in the database to support more than one table at a time being shrunk
- 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 non-leading columns

Shortcomings

- 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

Tools Available To Analyse Stats

- optdiag
- sp_showoptstats
- sp__optdiag
- sp_rpm_summ_stats

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
- It has to be run by a user with sa_role
  - and on a host with the same version of ASE installed as is used to host the target database

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 is broken in 16.0 SP03 PL02 on RHEL 7.4

sp_showoptstats

- SAP supply sp_showoptstats
- It was "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 has taken until ASE 16.0 SP03 to fix the bugs
- It only outputs the information in XML

sp_showoptstats (cont)

- 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

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 but did not update it for later versions
  - it has a few bugs
    - e.g. looping problems for all tables
  - it does not require sa_role to execute it
  - it has the same granularity as optdiag, but with wild cards and it can have a specific column for multiple tables

sp__optdiag (cont)

- I have updated it so that there are distinct versions for ASEs 15.5, 15.7 and 16.0
  - the bugs I was able to identify have been fixed
  - the output is exactly the same as that produced by optdiag for ASE 16.0 (apart for a few rounding differences)
    - the derived statistics returned by that function are sometimes different to those calculated by optdiag for ASEs 15.5 and 15.7
  - it can also output extra info I find to be of use
  - it is up to about four times faster than optdiag

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) 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)

- Supply "?" or "help" as the first parameter for information on its use and output
- 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
- 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 (but stay tuned)
    - use t if several "update index statistics {table}" are running at the same time for different tables (multiple connections)

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, n or n.m : meaning not in any index but has stats, not in any NCIs but in the CI, in 'n' NCIs, or in 'n' NCIs and is the first column in 'm' 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
    - it is calculated using datediff with this column's moddate and the closest previous moddate
      - but only for the same table if t is used instead of y

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 (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 summary

```
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)
  - a large 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

Cheating With Statistics

- 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 (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 (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 and 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 in a situation like this, once for each value that needs to be changed

```
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)

- Supply "?" or "help" as the first parameter to get information on how to use the procedure
- 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
- RangeAbsolute, RangeFactor, TotalAbsolute and TotalFactor use sp_modifystats to make changes and there must be existing statistics to change

sp_rpm_custom_stats (cont)

- Only ASE 15.7 ESD#2+ allows all of the settings above to be changed
  - earlier versions cannot change TuneFac, Sampling or Hashing
- The shortcoming of this system procedure is that statistics have to be updated after the settings have been customised
- However, it allows for a single stats updating script to be used, with no customisation needed
  - if columns do need different values, using this procedure is easier than changing scripts

sp_rpm_copy_stats

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

sp_rpm_copy_stats (cont)

- Consequently, the statistics on the existing data are still valid in the new version of the table
- I've written it 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 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 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

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)

- 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_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 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) 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)

- 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
- 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
  - 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 → c78 and c79 of this row

sp_rpm_shuffle_stats (cont)

- The pair of cells with the previous newest 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 this by further 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_append_stats

- If a 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

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 append_stats will only process the next date after the newest date with stats

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
  - append_stats was executed

```
sp_rpm_append_stats B_HIST, dt, 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)
```

    - A higher 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

sp_rpm_append_stats (cont)

- 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

- 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 append_stats will allow the optimiser to make better estimates between stats updating

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
  - we host the first 10 KM run of the year in Scotland, held on the 2nd of January every year (weather permitting)
  - it is called "The Lumphanan Detox 10K"
  - please get in touch if your firm would like to help sponsor a race
- `www.lumphanan.com/ase`

Summary

- A bit about myself
- How statistics are usually gathered
- Tools for analysis
- Customisation procedure
- Other ways of cheating
- Where to find the procedures