t_name	owner	rowcnt	col	ind_summ	moddate		pos_elap	req_step	act_step
CustInf	dbo	947308	AccountID	1C2N1L	Jun 22 2006	1:53:37:460PM	00:00:15	4000	4000
CustInf	dbo	947308	AccountType	0C1N1L	Jun 22 2006	1:56:14:640PM	00:00:13	4000	2602
CustInf	dbo	947308	BaseRate	0C3N1L	Jun 22 2006	1:56:01:640PM	00:00:09	20	1
CustInf	dbo	947308	CustomerID	0C5N2L	Jun 22 2006	1:55:43:640PM	00:01:04	4000	2267
CustInf	dbo	947308	CustomerLinkType	0C1N0L	Jun 22 2006	1:56:51:646PM	00:00:37	20	2
CustInf	dbo	947308	LastUpdate	0C3N1L	Jun 22 2006	1:55:52:640PM	00:00:09	4000	3006
CustInf	dbo	947308	StatusKey	0C2N2L	Jun 22 2006	1:54:39:546PM	00:01:02	4000	2995
CustInf	dbo	947308	StatusKeySize	0C1N0L	Jun 22 2006	1:57:34:723PM	00:00:43	20	11

Table 2

```
sp_rpm_stats_summ CustInf, NULL, Y, F, P
go
```

This generates a series of **sp_rpm_custom_stats** commands to be executed (the new command generated for the column in the example in Table 2.)

```
execute sp_rpm_custom_stats 'dbo.CustInf', CustomerID, '2730', NULL,
ReqStep, NULL, N  -- req = 4000 act = 2267
go
Changed requested steps for column CustomerID (ID = 2) of table
dbo.CustInf (ID = 1280004560) in sysstatistics for database CDI (ID = 4),
from 4000 to 2730
(return status = 0)
```

Running sp_rpm_stats_summ again gives us:

t_name	owner	rowcnt	col	ind_summ	moddate		pos_elap	req_step	act_step
CustInf	dbo	947308	AccountID	1C2N1L	Jun 22 2006	1:53:37:460PM	00:00:15	4000	4000
CustInf	dbo	947308	AccountType	0C1N1L	Jun 22 2006	1:56:14:640PM	00:00:13	3130	2602
CustInf	dbo	947308	BaseRate	0C3N1L	Jun 22 2006	1:56:01:640PM	00:00:09	20	1
CustInf	dbo	947308	CustomerID	0C5N2L	Jun 22 2006	1:55:43:640PM	00:01:04	2730	2267
CustInf	dbo	947308	CustomerLinkType	0C1N0L	Jun 22 2006	1:56:51:646PM	00:00:37	20	2
CustInf	dbo	947308	LastUpdate	0C3N1L	Jun 22 2006	1:55:52:640PM	00:00:09	4000	3006
CustInf	dbo	947308	StatusKey	0C2N2L	Jun 22 2006	1:54:39:546PM	00:01:02	4000	2995
CustInf	dbo	947308	StatusKeySize	0C1N0L	Jun 22 2006	1:57:34:723PM	00:00:43	20	11

Notice that only the requested step values have changed for two of the columns. All of the other information is the same.

Rolling Dates

Rolling dates is a proposed method of updating the statistics for a date column without actually performing **update [index] statistics**. It is aimed at an historical table that contains a set number of days' worth of data (say, 28 days), where each day contains roughly the same number of rows. During an overnight batch run, older data is deleted from the historical table and the current day's data is added from the working set. After the data has been rolled, this means that there is not actually any data for the oldest date stored in sysstatistics, and the newest date does not have an entry at all. If this disparity cannot be tolerated, the only way to resolve the issue is to update statistics for the column after the data has rolled. This is not possible in tables with many millions of rows and a small run window.

Rolling statistics would determine the oldest date for the column (in the first histogram cell), and, if there is no data, cells with that date are saved. The remaining cells are rolled, so that the next date would be in cells 1 and 2. The last two cells would still be the previous newest date, placed two cell positions down. The saved entries are modified to reflect the new date and their "inserted" position at the end of sysstatistics. This method relies on there being two histogram cells per date—so there must be at least ((number of dates to be stored * 2) + one) requested steps to ensure that the rolling of the statistics information always has a weight of 0 for the first cell and the weight of the actual date for the second cell.

Conclusion

The use of sp_rpm_stats_summ to summarize statistics for one or more tables in a database and sp_rpm_custom_stats to set requested step values (or distribution information) allows for the easier maintenance of statistics for individual columns. ■