OPTIMIZING STORAGE OF SAS DATA SETS

Martin J. Rosenberg and Carol A. Bogardus
Syntex Research

ABSTRACT

By default, the SAS® system stores numeric variables in a SAS data set in 8-byte representation. The LENGTH statement can be used to override the default and store numeric variables in a smaller space, thereby reducing storage costs. While this may be safely performed with integer valued variables, non-integer values will frequently be truncated leading to serious computational or logical errors. This paper presents an easy to use MACRO for identifying which variables in SAS data set are integer valued, allowing safe usage of the LENGTH statement.

I. INTRODUCTION

SAS data sets contain two kinds of variables: numeric and character. The length of a character variable is either explicitly set by a user in an INPUT or LENGTH statement, or is implicitly determined by SAS at the first occurrence of the variable. Once determined, the length of a character variable is the same both for manipulation within a DATA or PROC step and for storage on disk or tape of a SAS data set.

SAS performs all calculations on numeric variables in double precision. Hence, the length of all numeric variable when used in a DATA or PROC step is 8-bytes. However, for purposes of storage the LENGTH statement may be used to set the length of any or all numeric variables from 2 to 8 bytes. For integer data (whole numbers), the optimum storage length of a numeric variable is determined by knowing the maximum absolute value of the variable. (An absolute value of a number is that number without a sign. For example the absolute value of -200 equals the absolute value of +200, and the absolute value of -300 is greater than the absolute value of +100.) Table 1 shows the maximum absolute value of integer data that can be exactly represented in numeric SAS variables from 2-bytes to 8-bytes in length.

For non-integer data (fractions), however, the task of deciding the smallest numeric variable length that will exactly represent the data is not straightforward. This is because, in every number system, there are some fractions which cannot be exactly represented. For example, in our usual base 10 system (the decimal system), the number 1/3 cannot be written exactly. If we had 4 decimal places to represent 1/3 we would write it as 0.3333. If we had 6 decimal places instead, we would represent it as 0.333333. Similarly, in the binary (base 2) and hexadecimal (base 16) number systems used by IBM mainframe computers, some fractions cannot be represented exactly. The difficulty arises in that many numbers which can be represented exactly in decimal notation cannot be represented exactly in binary notation. Hence, if a LENGTH statement is used to reduce the length of a non-integer variable, the values stored in the SAS data set will be truncated and represented as accurately as the length permits. When the variable is used in a subsequent DATA or PROC step, its values will be expanded to 8-byte representation by padding with zeros. Hence, the expanded 8-byte representation will not be exactly equal to the original number in much the same way that 0.3333 does not exactly equal 0.333333 although both are the best representations of 1/3 in the space allowed. The program shown in Figure 1 illustrates the problems that may ensue.

FIGURE 1
Illustration of Potential Difficulties if Fractions are Truncated

DATA ONE;
    INPUT A B C;
    LENGTH A 4;
    CARDS;
    2.1 7.3 6
DATA TWO;
    IF A - 2 = 0.1;

In the first DATA step, data set ONE is created. ONE contains a single observation of
3 numeric variables, A, B, and C. Variable A is stored as a 4-byte variable while variables B and C default to 8-bytes. The values of A, B, and C are 2.1, 7.3, and 6, respectively. In the second DATA step, data set TWO is created by using a subsetting IF statement on data set ONE. However, even though 2.1 minus 2 does equal 0.1, TWO will have no observations. The subsetting IF statement compares a 4-byte representation of 0.1 on the left side of the equation, to an 8-byte representation on the right side of the equation. The two are not equal and so we wind up with the undesired result that data set TWO has no observations. Due to problems such as this we recommend that, unless minimizing storage is of prime importance, any numeric variables that contain fractions be stored in 8-bytes.

In large data processing operations where data are initially entered and stored in an external database and subsequently transferred to the SAS system for processing, it may not be immediately clear to the data analyst which numeric variables are integer and which are not. The purpose of this paper is to present an easy-to-use SAS macro which examines the variables in a SAS data set, determines which contain only integer data, and reports the minimum variable length needed to represent all the values exactly.

II. THE PROGRAM

The program is written as a macro named NVLENGTH and is shown in Figure 2. The MACRO requires three arguments to be specified in the following order: the name of the SAS data set to be checked, the name of the first variable to be checked, and the name of the last variable to be checked. In this way the MACRO can scan either all or only a subset of the variables in the dataset. The order of the variables can be obtained by running a PROC CONTENTS. The first data step (lines 11-18) scans the numeric variables. The second data step (lines 20-48) subsets out the numeric variables into an array called NVAR. Another array called DVAR is created which will flag values which are not integer valued. The INT function (line 38) is used to truncate the fractional part of each number. For example, INT(2.9) = 2. The original value of each numeric variable is compared to its truncated value. If the two are equal, the value is an integer and the flag variable DVAR is set to 0. Otherwise, the flag variable DVAR is set to 1, indicating the value was not integer. The INT function has a bug in it that causes INT to occasionally truncate negative numbers incorrectly. To get around this bug, the ABS (absolute value) function was used. This DATA step also determines the names of all the numeric variables by using the CALL VNAME function (line 39). The flag variables and the original data are output to the SAS data set VDATA. A set of variables whose values are the names of the numeric variables is output to the data set VNAMES.

In the third step of the MACRO (lines 51-55), PROC SUMMARY is used to sum each of the flag variables. The sum of each flag variable represents the number of values of the corresponding numeric variable which are not integers. Hence, if the sum equals zero, the corresponding numeric variable is integer valued. The PROC SUMMARY step also computes the minimum and maximum for each numeric variable. This will be used for integer variables to determine the minimum variable length. Lastly, a DATA NULL step (lines 66-107) is used to merge the names of the numeric variables with their minimums, maximums, and the sums of the flag variables. A report of the values of all numeric variables in data set ONE is generated. Additionally, a report for each numeric variable which are not integer valued, its minimum and maximum values, and the minimum variable length needed to represent exactly all values. Figure 3 displays a sample report.

To use this macro, one adds the appropriate control cards that define where the SAS libraries are located. The MACRO is called by statements of the form:

\[ \text{NVLENGTH (data set name, first variable, last variable);} \]

One such call is used for each data set to be scanned. The MACRO NVLENGTH only identifies which variables are integers. To then reduce the storage costs, one must include an appropriate LENGTH statement in any DATA step in which the SAS data set is being rewritten.

III. CONCLUSIONS

Storage costs of SAS data sets can be reduced by identifying which numeric variables contain only integer values and by using a LENGTH statement to reduce the length of these variables from the 8-byte SAS default. The MACRO NVLENGTH can be used to identify integer valued variables in data sets too large for visual inspection. The actual savings that will accrue are a function of the number of integer variables, the length of time the SAS data set is to be kept on disk, and the costing algorithms of your particular installation. Table 1 shows that a 4-byte variable can hold integers up to 16.7 million. Even a 2-byte variable holds integers up to 255. Thus, in many practical situations, the storage per integer variable can be at least halved. Note, however, that even if all variables in a SAS data set can be halved, due to the structure of a SAS data set, the length of the entire data set will not necessarily be cut in half. Nonetheless, the potential for savings is significant.

REFERENCES

**FIGURE 2**

---

1. **MACRO TO LIST INTEGER NUMERIC VARIABLES AND THE LENGTH ☑️**
2. **IN BYTES NECESSARY TO STORE THEIR VALUES. ☑️**
3. **FILE** REFERS TO THE SAS FILE NAME, ☑️
4. **'NVARF'** REFERS TO THE FIRST NUMERIC VARIABLE NAME, ☑️
5. **'NVARL'** REFERS TO THE LAST NUMERIC VARIABLE NAME. ☑️

---

**MACRO NVLENGTH(FILE,NVARF,NVARL):** ☑️

---

1. %MACRO NVLENGTH(FILE,NVARF,NVARL);
2. DATA NVARDATA;
3. OPTIONS OBS=1;
4. SET &FILE;
5. COUNT THE NUMBER OF NUMERIC VARIABLES BETWEEN AND ☑️
6. INCLUDING NVARF AND NVARL.
7. N=NMISS(OF &NVARF-NUMERIC-&NVARL)+N(OF &NVARF-NUMERIC-&NVARL)
8. CALL SYMPUT ('NV',LEFT(N));
9. DATA VDATA (DROP=NAME1-NAME&NV)
10. VNAMES (KEEP=NAME1-NAME&NV);
11. OPTIONS OBS=MAX;
12. SET &FILE;
13. ARRAY OF NUMERIC VARIABLE NAMES. ☑️
14. ARRAY NAME $ 8 NAME1-NAME&NV;
15. *** ARRAY OF DIFFERENCE SUMS. ☑️
16. ARRAY DVAR DVARI-DVAR&NV;
17. *** ARRAY OF NUMERIC VARIABLE VALUES. ☑️
18. ARRAY NVAR &NVARF-NUMERIC-&NVARL;
19. *** ARRAY OF NUMERIC VARIABLE VALUE DIFFERENCES. ☑️
20. CALL SYMPUT ('V',&N);
21. DATA VNAMES (KEEP=NAME1-NAME&NV);
22. OPTIONS OBS-MAX;
23. SET FILE;
24. *** ARRAY OF NUMERIC VARIABLE NAMES. ☑️
25. ARRAY NAME $ 8 NAME1-NAME&NV;
26. *** ARRAY OF MINIMUM NUMERIC VARIABLE VALUES. ☑️
27. ARRAY VMIN VMINI-VMIN&NV;
28. *** ARRAY OF MAXIMUM NUMERIC VARIABLE VALUES. ☑️
29. ARRAY VMAX VMAXI-VMAX&NV;
30. *** ARRAY OF MAXIMUM LENGTH NECESSARY TO STORE ☑️
31. NUMERIC VARIABLES WHOSE FLAG SUMS EQUAL ☑️
32. DO OVER I:
33. * MACRO NVLENGTH FILE,NVARF,NVARL;
34. DATA _NULL_; ☑️
35. OPTIONS NOOS;
36. FILE PRINT NOTITLES PAGE SIZE=60 HEADER=TITLES N=PS;
37. *** ARRAY OF NUMERIC VARIABLE NAMES. ☑️
38. ARRAY NAME $ 8 NAME1-NAME&NV;
39. *** ARRAY OF DIFFERENCE SUMS. ☑️
40. ARRAY DSUM DSUM1-DSUM&NV;
41. *** ARRAY OF MINIMUM NUMERIC VARIABLE VALUES. ☑️
42. ARRAY VMIN VMINI-VMIN&NV;
43. *** ARRAY OF MAXIMUM NUMERIC VARIABLE VALUES. ☑️
44. ARRAY VMAX VMAXI-VMAX&NV;
45. PROC FORMAT;
46. VALUE LENGTH 0-255 = '2'
47. 256-65535 = '3'
48. 65536-16777215 = '4'
49. 16777216-4294967295 = '5'
50. 4294967296-1099511627775 = '6'
51. 1099511627776-281474946710656 = '7'
52. 281474946710656-HIGH = '8'
53. DATA NULL;
54. MERGE LESUMMS VNAMES;
55. FILE PRINT NOTITLES PAGE SIZE=60 HEADER=TITLES N=PS;
56. *** ARRAY OF NUMERIC VARIABLE NAMES. ☑️
57. ARRAY NAME $ 8 NAME1-NAME&NV;
58. *** ARRAY OF DIFFERENCE SUMS. ☑️
59. ARRAY DSUM DSUM1-DSUM&NV;
60. *** ARRAY OF MINIMUM NUMERIC VARIABLE VALUES. ☑️
61. ARRAY VMIN VMINI-VMIN&NV;
62. *** ARRAY OF MAXIMUM NUMERIC VARIABLE VALUES. ☑️
63. ARRAY VMAX VMAXI-VMAX&NV;
64. PROC SUMMARY DATA=VDATA;
65. VAR &NVARF-NUMERIC-&NVARL DVARI-DVAR&NV;
66. OUTPUT OUT=LESUMMS SUM(DVAR1-DVAR&NV)=DSUM1-DSUM&NV
67. MIN(&NVARF-NUMERIC-&NVARL)=VMINI-VMIN&NV
68. MAX(&NVARF-NUMERIC-&NVARL)=VMAXI-VMAX&NV;
69. PROCSUM DATA=VDATA;
70. TITLE: MAINTITL='TABLE LISTING OF INTEGER NUMERIC VARIABLES';
71. CENTER=INT(80-LENGTH(MAINTITL))/2.01
72. PUT #8 @CENTER MAINTITL
73. #12 @10 8*-'
74. #25 @25 6*-'
75. #38 @38 13*-'
76. #58 @58 13*-';
77. RETURN;
78. %MEND NVLENGTH;
79. %NVLENGTH (FILE,NVARF,NVARL);
80. *** MACRO NVLENGTH FILE,NVARF,NVARL;
81. *** MACRO TO LIST INTEGER NUMERIC VARIABLES AND THE LENGTH ☑️
82. *** IN BYTES NECESSARY TO STORE THEIR VALUES. ☑️
83. *** FILE** REFERS TO THE SAS FILE NAME, ☑️
84. *** 'NVARF'** REFERS TO THE FIRST NUMERIC VARIABLE NAME, ☑️
85. *** 'NVARL'** REFERS TO THE LAST NUMERIC VARIABLE NAME. ☑️
86. *** MACRO NVLENGTH FILE,NVARF,NVARL;
## TABLE LISTING OF INTEGER NUMERIC VARIABLES

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>LENGTH</th>
<th>MINIMUM VALUE</th>
<th>MAXIMUM VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDY</td>
<td>3</td>
<td>903</td>
<td>903</td>
</tr>
<tr>
<td>VISDATE</td>
<td>3</td>
<td>8130</td>
<td>8873</td>
</tr>
<tr>
<td>INV</td>
<td>3</td>
<td>1808</td>
<td>1811</td>
</tr>
<tr>
<td>STARTDATE</td>
<td>3</td>
<td>8125</td>
<td>8858</td>
</tr>
<tr>
<td>DATEB</td>
<td>3</td>
<td>8040</td>
<td>8873</td>
</tr>
<tr>
<td>EVENT</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>NITROCNT</td>
<td>2</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>ENDDATE</td>
<td>3</td>
<td>8130</td>
<td>8873</td>
</tr>
<tr>
<td>FOUND</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

For more information contact the authors at:

Syntex Research A3-434
P.O. Box 10850
Palo Alto, CA 94303

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA.