Data Entry Using Extended Tables in Release 6.07 of the SAS® System

Allan M. Kominek
Euclid, Ohio

ABSTRACT

This paper discusses a system for entering the results of a survey. The information for this particular survey relates to the background of people in the Cleveland SAS Users Group. Through the use of SAS/AF®, Screen Control Language (SCL), and extended tables, a system was developed that displayed the question and all possible choices, including blank lines for comments. The data entry person could then select the answers from the list and enter any comments. The entire system required one screen and three SAS data sets.

BACKGROUND

As president of the Cleveland SAS Users Group, one of my main concerns has been how do I keep in touch with the general membership? Our membership consists of over 200 hundred people from over 50 companies across northeastern Ohio. It is awfully difficult to talk to everyone individually. So, invariably, someone mentions the dreaded word SURVEY! Well, after the cringing has subsided, and some semblance of sanity has settled over those gathered to discuss plans for the users group, the general discussion begins:

Remember the last time? We sent out 225 surveys and got back 85.

Hey, count your blessings, you didn't have to enter all that data into that PROC FSEDIT screen. It took forever!

Yes, now it's all becoming clear. The worst part wasn't the 30% response, but trying to enter those eighty odd surveys into SAS so that PROC FREQ's could be run. The first attempt to enter the data involved trying to type the actual answers in free form. That was O.K. as long as everything was spelled consistently, which it wasn't. The next brilliant idea involved assigning codes to each of the possible answers and just typing the codes in. That seemed easy enough in concept. It was putting it into practice that wasn't so easy. The codes were assigned on a blank survey. It would have been too time consuming to go through and hand code all the existing surveys. All the data entry person had to do was look at the actual survey and then look at the coded survey to determine what code to enter. This is the reason that the president of the users group sometimes has trouble keeping his eyes from crossing. No wonder the word survey is usually accompanied by all that cringing. The alternative would have been to put a little forethought into

the survey and design it with the coding already on it, i.e., choose a,b,or c. Even in this case, it is sometimes very easy to lose one's place.

Did I just enter question number 15 or 16?

Not only does one suffer from eye crossing, but now one is often accused of talking to oneself!

ANOTHER METHOD of DATA ENTRY

With 200 members, there is no way that one can avoid the "S" word. So, how to avoid the cringing, the eye crossing, and the talking to oneself? Eliminate the cause of the cringing! Eliminate the data entry nightmare!

In thinking about how to simplify the entry of any survey, the idea of displaying a blank survey on-line and duplicating each respondent's answers came up. This sounded rather messy until SCL and dynamic extended tables came into the picture. By using an extended table as a selection list, all potential answers can be displayed on the screen and the data entry person can select the answer or answers by cursoring to them and pressing enter. To further simplify the process, the entire question can also be displayed above the answers. It's very difficult to become confused when everything is on the screen in front of you.

GETTING STARTED

Definitely, the place to start is to construct the screen. The reader is referred to Figure 1 at the end of this paper. The first point that should be explained is that when using extended tables, the screen is divided into two distinct areas, the non-scrollable area and the scrollable area. The extended table is contained in the scrollable area. The scrollable area is separated from the non-scrollable area by the three caret symbols (^^^). (What symbol is used may depend on your system and your keyboard. The reader is referred to SAS Screen Control Language: Reference, Version 6, First Edition for the other characters that are available.) The reader will also notice that only one line has been defined for the extended table.

The next area of concentration is the attributes. There are two attributes that must be set in order for this application to work properly. The first one is in the General Attributes (GATTR) Screen. For System Options, specify EXTENDED TABLE. The second attribute is in the Field Attributes Screen. A Choice group must be specified for the field in the scrollable area.

Once the attributes have been set, the actual SCL program can be written. The reader is referred to Figure 2, for the exact contents of the entire program. The following is a brief discussion of each of the major sections of the program.

The INIT Section

The main feature of this section is that it defines the extended table for the first

time.

```
CALL SETROW(0,25,'N','Y');
```

The first argument indicates the maximum number of rows for the table. Since the number of rows will depend on the number of possible answers, there really is no known maximum number. This is indicated by the 0, which means that the table will be dynamic, changing with the number of answers available. The next argument indicates the maximum number of selections allowed. This was arbitrarily set at 25. The third argument is the selection order. N specifies that selected items are highlighted, but are not moved to the top of the list. The list remains in its original order. The final argument indicates that the table is dynamic. The screen control variable quest is initialized to 001, which is the first question number in this application.

The GETROW Section

This is the section executed immediately after the INIT section. There are three permanent data sets involved in the application. The first three statements close any of these data sets that may be open. Following these three statements is a submit block. This submit block selects the question text from the question data set and selects the available answers from the answer data set. The question data set has four variables, one variable contains question number and the other three contain the question text. The answer data set contains two variables, one for question number and one for an answer. In most cases, there will be more than one observation (answer) per question. Remember that upon first entering the application, quest was set to 001. So, initially, the user will see question 001 on the screen. The next four statements make sure that something has been selected by the submit block. The user may have enter an invalid question number. The question and answers selected are stored in temporary data sets. The two temporary data sets are opened for input. The third permanent data set, the survey data set, will be opened for update. It will contain the results of the survey. A CALL SET is done for these three data sets. This ensures that the question and answer text will appear on the screen and that the data selected will eventually make it into the survey data set. The question and answers are then fetched from the temporary data sets. A PF key has been defined as NXTQUEST (Next Question).

The MAIN Section

The system variable, _msg_, is initialized to blank. The screen control variable WORD is set equal to the first command issued using the WORD() function. If the Next Question PF key was pressed, the current question number is incremented by one. Remember, the MAIN section executes before the GETROW section. If the user enters another question number or presses the Next Question PF key, the next section of code executes.

Notice that another SETROW is executed. This blanks out any previous answers that were displayed. This was done because in some instances where a question with a large number of answers was selected before one with fewer answers, the extra answers from the previous question would scroll by and then disappear. This could be rather unnerving to the inexperienced user as several experienced users were observed leaping out of their chairs at this phenomenon. The question dataset is closed if it is open. The new question is then placed in a temporary data set and fetched to the screen.

The PUTROW Section

The answers selected by the user are appended to the survey data set.

The TERM Section

Upon exiting, all open data sets are closed. The survey data set is sorted by survey id and question number. Duplicate observations are eliminated.

Conclusion

Although for this one application, the time spent perfecting the idea may not have totally been equalled by the time saved during the data entry phase, the time saved for other applications will more than make up for it. The knowledge gained regarding extended tables will be useful in other applications that require selection lists.

Contacting the Author

Allan M. Kominek
315 East 216th Street
Euclid, OH 44123
(216) 479-1260 x6047

SAS and SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries, ® indicates USA registration.

Figure 1

```
Survey Number  : &__

Question Number: &__

&QUESTXT1_____________________________________________________________

&QUESTXT2_____________________________________________________________

&QUESTXT3_____________________________________________________________

^^^

&ANSWER_______________________________________________________________
```

Figure 2

```
INIT:
   CONTROL ASIS; CONTROL ALWAYS;
   CALL SETROW(0,25,'N','Y');
   DSID1=0; DSID2=0; DSID3=0;
   QUEST='001';
RETURN;
MAIN:
   _MSG_=''; WORD=WORD(1);
   IF WORD='NXTQUEST' THEN QUEST=PUT(INPUT(QUEST,3.)+1.Z3.);
   IF MODIFIED(QUEST) OR WORD='NXTQUEST' THEN DO;
      CALL SETROW(0,25,'N','Y');
      IF DSID2 NE 0 THEN DSID2=CLOSE(DSID2);
      SUBMIT CONTINUE;
         DATA QUEST; SET SURV.QUEST(WHERE=(QUEST="&QUEST")); RUN;
      ENDSUBMIT;
      DSID2=OPEN('QUEST','I');
      CALL SET(DSID2);
      RC=FETCHOBS(DSID2,1);
   END;
RETURN;
TERM:
   IF DSID1 NE 0 THEN CLOSE(DSID1);
   IF DSID2 NE 0 THEN CLOSE(DSID2);
   IF DSID3 NE 0 THEN CLOSE(DSID3);
   SUBMIT CONTINUE;
      PROC SORT DATA=SURV.SURVEY NODUPS; BY SURV_ID QUEST; RUN;
   ENDSUBMIT;
RETURN;
GETROW:
   IF DSID1 NE 0 THEN CLOSE(DSID1);
   IF DSID2 NE 0 THEN CLOSE(DSID2);
   IF DSID3 NE 0 THEN CLOSE(DSID3);
   SUBMIT CONTINUE;
      DATA ANS; SET SURV.ANS(WHERE=(QUEST="&QUEST"));
      DATA QUEST; SET SURV.QUEST(WHERE=(QUEST="&QUEST")); RUN;
   ENDSUBMIT;
   DSID1=OPEN('ANS','I');
   IF DSID1 LE 0 THEN DO;
      _MSG_='No Such Question.';
     RETURN;
   END;
   DSID2=OPEN('QUEST','I');
   DSID3=OPEN('SURV.SURVEY','U');
   CALL SET DSID1; CALL SET DSID2; CALL SET DSID3;
   RC=FETCHOBS(DSID2,1);
   IF FETCHOBS(DSID1,_CURROW_) THEN DO;
      IF WORD='NXTQUEST' THEN WORD='';
      CURSOR ANS;
      CALL ENDTABLE();
   END;
RETURN;
PUTROW:
   RC=APPEND(DSID3);
RETURN;
```