SAS REPORT GENERATOR: A SAS PROGRAM TO GENERATE SAS CODE

R L Anderson and M P Koranda
IBM System Products Division

ABSTRACT

This paper presents a high level interface program that writes SAS code allowing users easy access to SAS's full capabilities and freeing the programmer from the chore of writing code to input data.

Programming in a group that supports a large number of users, each accessing varying amounts of data and requiring different forms of data analysis, presents many problems. Some of the difficulties encountered are: interfacing with users ranging from nonprogrammers to highly experienced programmers, requests for one-time-only reports, the need for minor report modifications stemming from users wanting to look at the same data from different perspectives, and the problems associated with the duplication of code among users. A description of the high level interface and an explanation of the data tables required, along with examples of the SAS code generated, will be presented.

Introduction

The purpose of the SAS Report Generator is to allow a common means for executing application reports, to simplify the writing of new reports, to minimize duplication of code in reports, and to give nonprogrammers the ability to easily produce reports.

A solution that meets these objectives is a two-step procedure. The first step reads user-entered parameters along with data tables and generates SAS code. The first step also generates a header page containing user information. The second step executes the code written in the first step.

The SAS Report Generator (SRG) can execute either prewritten reports or dynamic reports (reports written by the driver at the time of execution).

The SRG, whether it is executing prewritten reports or dynamic reports, writes the SAS code (in the form of a SAS macro) to be used to input the needed data. The code that is written will input only those variables which are needed by the requested report.

In order to write this code, the SRG must have access to the data definition tables. These tables contain the name, the position within the record, the data type and length of each variable. There is a data definition table for each data set defined to the SRG. (A complete description of the data definition tables will be presented later.) The tables are the only places where the data is defined. Every report that needs data uses these tables. Therefore, if the data sets ever change, the data definition tables are the only things that must be changed for the SRG.

The SAS Report Generator determines which variables are needed by looking at both the user input and a report variable data set. The user's input data is entered using a high-level keyword language. For a dynamic report, all the variables needed must be listed on the user input. If a prewritten report is requested, only those variables which are not normally used by that report need be listed. The SRG will obtain a list of the other variables from the report variable data set. By using the report variable data set, the SRG removes this responsibility of the user for keying in all the needed variables on the user input.

The user input also contains any constraints that may be leveled against the input data. If any range of values or discrete list of values is desired, it is through the user input that the SAS Report Generator is informed. The SRG writes the code necessary to check the input data to determine whether it qualifies. This code is written at the same time as the input code is written and becomes a part of the SAS input macro. The first variable that does not qualify will cause the input data record to be deleted. None of the remaining variables will be read in or checked for that input record.

Now that all the code has been written to input and qualify the data, the SRG must obtain the remainder of the code necessary for the requested report. If it is a prewritten report, all that is necessary is to include the code for that report. Otherwise, the SRG itself must write all the code. When all the code is obtained, the requested report is executable. The last thing the SAS Report Generator does is execute the report.

The SAS Report Generator executes prewritten and dynamic reports. The prewritten reports are those whose
outputs are static. Only the data which they run against changes. An example of a prewritten report would be one to calculate yields from data records containing quantities of good parts and quantities of scrapped parts. The SRG would write input code and call in the prewritten code. This prewritten code would use the newly written input code to input the data. The prewritten code would then perform the necessary yield calculations and output the results. Prewritten reports are available for any data analysis request which is too complicated for the dynamic reports. The dynamic reports are not prewritten. They are written by the SAS Report Generator during execution, thereby resulting in the user having control over the output. Dynamic reports execute either one SAS PROC or a PROC SORT and one other SAS PROC. An example would be a simple sorted listing of data. The SAS Report Generator would write code to input the data into a SAS data set, sort the data using PROC SORT, and print a listing using PROC PRINT. Complete data analysis may be performed to the users' specifications by executing several dynamic reports in various combinations.

The SAS Report Generator builds a header page. This page, listed before each report, contains the name of the report in large letters, a list of the code the user entered to generate the report, and a bulletin board section for passing information and messages to the users.

The user enters his input data via a keyword language. There are three groups of keywords: the first specifies what type of report to produce, the second allows the user to specify what data and how much of the data to use, and the third group of keywords allows the reports to be customized for each individual user.

The report keywords allow the user to specify what report to produce. For a prewritten report, the user enters only the name of the prewritten code. If the user is producing a dynamic report, the user enters the name of the SAS PROC and any options and values desired for that PROC.

The SAS Report Generator uses the data-related keywords in conjunction with the data definition tables to write input and qualification code. The user enters where the data is stored, what variables to use, and how to qualify the variables. Any variable may be qualified. The user can request a range of values or a discrete list of values for a variable. The SAS Report Generator will build code to read in the data and keep the observation only if all variables pass the qualifications placed on them. For example, assume you have data with a variable called DATE. If you wanted a report using data from certain days, you would use the keyword as follows:

```
DISCRETE DATE 820104 820105 820106
```

The SRG would look up in a data definition table how to input DATE. It would build the following code:

```sas
INPUT @ 9 DATE $6. @;
IF DATE = '820104'
| DATE = '820105'
| DATE = '820106';
```

The variable DATE would be read in and only records with the proper data values would be kept.

The user-customization keywords pertain mostly to the dynamic reports. They allow users to put titles on the report. Through these type of keywords, users can add code to the program the SAS Report Generator writes allowing users much more flexibility in the dynamic reports they are able to produce.

There is one data definition table for each type of data defined to the SAS Report Generator. When a user requests a variable, the SRG obtains the information explaining how to input the variable from the proper definition table. Assume a labor data record with the layout shown in Figure 1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Format</th>
<th>Len</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee</td>
<td>ebcdic</td>
<td>10</td>
<td>name of person who worked the parts</td>
</tr>
<tr>
<td>date</td>
<td>ebcdic</td>
<td>6</td>
<td>date parts were worked (YYYYMMDD)</td>
</tr>
<tr>
<td>hours</td>
<td>ebcdic</td>
<td>4</td>
<td>amount of time spent working the parts</td>
</tr>
<tr>
<td>worked</td>
<td>ebcdic</td>
<td>4</td>
<td>number of parts worked</td>
</tr>
<tr>
<td>good</td>
<td>ebcdic</td>
<td>4</td>
<td>number of good parts</td>
</tr>
<tr>
<td>part</td>
<td>ebcdic</td>
<td>4</td>
<td>part number</td>
</tr>
<tr>
<td>flags</td>
<td>binary</td>
<td>1</td>
<td>parts are being reworked</td>
</tr>
<tr>
<td>rwork</td>
<td></td>
<td></td>
<td>parts are part of an experiment</td>
</tr>
</tbody>
</table>

![Figure 1. Sample Labor Data Layout](image)

Table 1 shows the data definition for this (Figure 1) data.

'Name' is the name given to the variable. 'Position' is the variable's position within the record. 'Type' identifies the format the data is stored in. This field
Table 1. Data Definition for Sample Labor Data

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Type</th>
<th>Length</th>
<th>Bit</th>
<th>Low Limit</th>
<th>High Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee</td>
<td>1</td>
<td>c</td>
<td>8</td>
<td>x</td>
<td>xxxxxxxxx</td>
<td>xxxxxxxxx</td>
</tr>
<tr>
<td>date</td>
<td>9</td>
<td>c</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hours</td>
<td>15</td>
<td>n</td>
<td>4</td>
<td></td>
<td>0000</td>
<td>8999</td>
</tr>
<tr>
<td>worked</td>
<td>19</td>
<td>n</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>23</td>
<td>n</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>part</td>
<td>27</td>
<td>c</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rework</td>
<td>31</td>
<td>p</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>exp</td>
<td>31</td>
<td>p</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

is used to determine what SAS input format to use to input the data. 
Length is the length of the variable. 
If the variable is a bit, the 'bit' field tells what bit location in the byte. 'Low limit' and 'high limit' perform two functions. If the variable is not a bit, it is the default high and low limits. 
When this variable is requested, code will be generated to make sure the value is in the default range. It can be overwritten by the user. In this example, part numbers in the 9000-9999 range are reserved for debug use. These would not normally be used in the reports. The defaults are used to free the users from having to enter data to qualify a majority of the times they use this variable. If the variable is a bit and if the bit is off, then the variable will be set to the low limit value; otherwise the bit variable will be set to the high limit value.

EXAMPLES

Following are examples of report requests. The user-entered parameters, the code generated by the SAS Report Generator, and the report output will be presented. Figure 2 contains sample labor data.

Figure 2. Sample Labor Data

This first example shows what happens when a user requests a prewritten report. The following prewritten code is what the user wishes to execute. The report name as shown in Figure 3, is YIELDRPT.

**INPUT THE DATA.**

**DATA IN:**

**INFILE**

**LABORDD ;**

**LABOR DD CARD.**

**SRG WRITTEN MACRO.**

**SORT THE DATA BY DATE.**

**PROC SORT**

**BY DATE**

**SUM UP THE QUANTITIES BY DAY.**

**PROC MEANS MAXDEC=2 NOPRINT**

**BY DATE**

**OUTPUT WORKED GOOD HOURS**

**SUM WORKED GOOD HOURS**

**PRINT THE DATA.**

**DATA _NULL ;**

**FILE PRINT HEADER=NEWPAGE ;**

**SET MOUT ;**

**YIELD = (GOOD/WORKED) * 100 ;**

**CALCULATE YIELD.**

**HF100P = (WORKED/HOURS) * 100 ;**

**CAL HOURS/100 PARTS.**

**PRINT THE DATA. ;**

(continued)
* CODE TO BUILD THE HEADINGS FOR A NEW PAGE.

PUT @ 50 'LABOR YIELD REPORT' // @ 5 'DATE' @ 15 'WORK' @ 25 'GOOD' @ 35 'YIELD' @ 45 'HOURS' @ 55 'HOURS/100 PARTS' / @ 5 '~~' @ 15 '~~' @ 25 '~~' @ 35 '~~' @ 45 '~~' :==: @ 55 '~~' , / ;
RETURN ;

Figure 3. Prewritten Report YIELDRPT

The report variable data set contains an entry for this report as follows:

YIELDRPT LABOR DATE HOURS WORKED GOOD END

It contains the report name, YIELDRPT, the data name, LABOR, and the list of variables to input. To request this report, a user would enter:

REPORT YIELDRPT END

The keyword REPORT tells the SAS Report Generator that a report name follows. The keyword END tells the SRG that this is the end of user-entered input data for this report.

First, the SAS Report Generator reads the input and builds the header page. The header page contains the report name in large block letters. It also contains the exact user-entered data lines so the user knows what he specified when he ran this report and could duplicate it if necessary. There is also a bulletin board area for conveying messages and information to users.

Table 2. Labor Yield Report

<table>
<thead>
<tr>
<th>DATE</th>
<th>WORK</th>
<th>GOOD</th>
<th>YIELD</th>
<th>HOURS</th>
<th>HOURS/100 PARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>820104</td>
<td>180</td>
<td>144</td>
<td>80.00</td>
<td>9.1</td>
<td>5.3</td>
</tr>
<tr>
<td>820105</td>
<td>198</td>
<td>153</td>
<td>77.27</td>
<td>8.4</td>
<td>5.5</td>
</tr>
<tr>
<td>820106</td>
<td>237</td>
<td>200</td>
<td>84.39</td>
<td>8.6</td>
<td>4.3</td>
</tr>
<tr>
<td>820107</td>
<td>236</td>
<td>189</td>
<td>80.08</td>
<td>12.5</td>
<td>6.6</td>
</tr>
<tr>
<td>820108</td>
<td>196</td>
<td>152</td>
<td>77.55</td>
<td>8.9</td>
<td>5.8</td>
</tr>
</tbody>
</table>

The SAS Report Generator uses this input, along with the report variable data set and the data definition table, to build the following code (Figure 4):

MACRO INSERT1
% MACRO INSERT2
% MACRO INSERT3
%
MACRO劳动 I
INPUT @ 9 'DATE $6.
@ 15 'HOURS 4.
@ 19 'WORKED 4.
@ 23 'GOOD 4.
; INSERT3
%
INSERT1
% INCLUDE REPORTS(YIELDRPT) ;
INSERT2
Figure 4. SAS Report Generator Written Code.

The macros INSERT1, INSERT2, and INSERT3 can contain actual lines of SAS code entered by the user. Macro LABOR I is the input macro. When executed, the report would produce the output shown in Figure 5 and Table 2.

Table 2. Labor Yield Report

<table>
<thead>
<tr>
<th>DATE</th>
<th>WORK</th>
<th>GOOD</th>
<th>YIELD</th>
<th>HOURS</th>
<th>HOURS/100 PARTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>820104</td>
<td>180</td>
<td>144</td>
<td>80.00</td>
<td>9.1</td>
<td>5.3</td>
</tr>
<tr>
<td>820105</td>
<td>198</td>
<td>153</td>
<td>77.27</td>
<td>8.4</td>
<td>5.5</td>
</tr>
<tr>
<td>820106</td>
<td>237</td>
<td>200</td>
<td>84.39</td>
<td>8.6</td>
<td>4.3</td>
</tr>
<tr>
<td>820107</td>
<td>236</td>
<td>189</td>
<td>80.08</td>
<td>12.5</td>
<td>6.6</td>
</tr>
<tr>
<td>820108</td>
<td>196</td>
<td>152</td>
<td>77.55</td>
<td>8.9</td>
<td>5.8</td>
</tr>
</tbody>
</table>

450
Suppose a user desired to produce a listing of the labor claims for January 4th, 5th, and 6th titled 'LIST OF LABOR CLAIMS ON JANUARY 4TH 5TH AND 6TH'. The user wanted the totals good, work, experimental, and reworked for each day and a summary at the bottom. The listing should be double spaced. The report request would look as follows:

REPORT PRINT
VAR LABOR EMPLOYEE DATE HOURS
WORKED GOOD PART REWORK EXP
DISCRETE DATES 810104 810105 810106
TITLE LIST OF LABOR CLAIM FOR
JANUARY 4TH 5TH AND 6TH
SUM WORKED GOOD RWKGOOD EXPGOOD
SUMBY DATE
RPTOPTS DOUBLE
INSERT3 IF REWORK = 1
INSERT3 THEN RWKGOOD = GOOD
INSERT 3 ELSE RWKGOOD = 0
INSERT 3 IF EXP = 1
INSERT3 THEN EXPGOOD = GOOD
INSERT 3 ELSE EXPGOOD = 0
END

REPORT PRINT tells the SAS Report Generator to execute a PROC PRINT. The variables to be printed follow the keyword VAR. The keyword title allows the title to be entered. SUM and SUMBY tell which variables to compute sums for and also specify how the data must be sorted. RPTOPTS is the keyword used to specify any of the PROC's options. The code following INSERT3 will be placed inside the SRG-written input macro. This allows the variables RWKGOOD and EXPGOOD to be created and summed.

First, the SAS Report Generator reads the input and builds the two separator pages. The SAS Report Generator would then use this input, along with the data definition table, and build the code shown in Figure 6.

The macros INSERT1, INSERT2, and INSERT3 can contain actual lines of SAS code entered by the user. Macro LABOR_I is the input macro. When executed, this report would produce this output (Figure 7).
LIST OF LABOR CLAIMS FOR JANUARY 4TH, 5TH, AND 6TH

-----------------------------------DATE=820104-----------------------------------

<table>
<thead>
<tr>
<th>OBS</th>
<th>EMPLOYEE</th>
<th>HOURS</th>
<th>WORKED</th>
<th>GOOD</th>
<th>PART</th>
<th>REWORK</th>
<th>EXP</th>
<th>RWKGOOD</th>
<th>EXPGOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SMITH</td>
<td>5.1</td>
<td>100</td>
<td>75</td>
<td>1234</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>JONES</td>
<td>4.0</td>
<td>80</td>
<td>69</td>
<td>1234</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

-----------------------------------DATE=820105-----------------------------------

<table>
<thead>
<tr>
<th>OBS</th>
<th>EMPLOYEE</th>
<th>HOURS</th>
<th>WORKED</th>
<th>GOOD</th>
<th>PART</th>
<th>REWORK</th>
<th>EXP</th>
<th>RWKGOOD</th>
<th>EXPGOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>SMITH</td>
<td>4.7</td>
<td>130</td>
<td>87</td>
<td>1234</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>JONES</td>
<td>3.7</td>
<td>68</td>
<td>56</td>
<td>1234</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

-----------------------------------DATE=820106-----------------------------------

<table>
<thead>
<tr>
<th>OBS</th>
<th>EMPLOYEE</th>
<th>HOURS</th>
<th>WORKED</th>
<th>GOOD</th>
<th>PART</th>
<th>REWORK</th>
<th>EXP</th>
<th>RWKGOOD</th>
<th>EXPGOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>SMITH</td>
<td>4.2</td>
<td>20</td>
<td>75</td>
<td>1234</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>SMITH</td>
<td>2.1</td>
<td>51</td>
<td>47</td>
<td>1234</td>
<td>1</td>
<td>0</td>
<td>47</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>JONES</td>
<td>6.2</td>
<td>164</td>
<td>142</td>
<td>1234</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 7: Dynamic PRINT report output

Conclusions

Implementing the SAS Report Generator results in a common means of executing application reports. Because the SRG writes SAS input code, this code is not duplicated in any reports using the same data. With the SRG's parameter passing and qualification capabilities comes flexibility in reports and eliminates reports whose only difference is what data is used. Through the keyword language and data definition tables, data analysis can easily be performed by users with minimal programming knowledge. Being able to add code to the dynamic reports gives a user the ability to customize reports and produce useful results quickly. The SAS Report Generator is a useful tool in controlling application reports.
INTERACTIVE TECHNIQUES

Session Leader:
Deborah Stewart, Radian Corporation

Moderator:
Sheryl Preston, Mitre Corporation