PhUSE US Connect 2019

Paper CT08

Can SAS programs have memory?

Rakesh Kumar, GlaxoSmithKline, Collegeville, USA
Joanne Zhou, GlaxoSmithKline, Collegeville, USA
David Wade, GlaxoSmithKline, Collegeville, USA

ABSTRACT

Memory is the ability to keep a record of information and an ability to access it when needed. Computers have memory. The mechanism by which a computer can remember is by storing data either temporarily in RAM or permanently on ROM or other internal or external storage devices and access it when needed. While RAM is used only while the computer is running, permanent memory is written either at the factory or during the execution of a program and does not erase when the machine is switched off. Computer programs are a set of instructions for performing a specific task, which when invoked, executes these instructions and outputs the result. This paper presents a way to exploit these features of computer and computer programs to bestow memory to SAS programs while performing a simple task of moving data from one source to another and being able to tell the differences in a sequential instream data without the need to store the older copy of the same data. This technique is extensible to many other applications as well.

INTRODUCTION

Memory is a fascinating concept. From living forms to computer, this unique ability to recall past events bestows them with unique set of abilities that would not be possible otherwise. The underlying principle of course is the ability to keep a record of information stored and an ability to access it when needed. Computers and other electronic gadgets achieve this by storing information on either internal or external storage device and access them when needed, during program execution. This got us thinking whether the same concept can be implemented to do some of the house keeping jobs that programmers routinely do. One such repetitive task is to copy sequential data from the data management (DM) staging area to the analysis and reporting (AR) area during a trial and be able to keep track of the changes in the data and provide feedback to the data management group. In this paper we demonstrate an application of a SAS programming technique to storing data attributes to serve as memory and then retrieving the same in the successive runs to output informative messages about data when executing without the need of storing previous data versions.

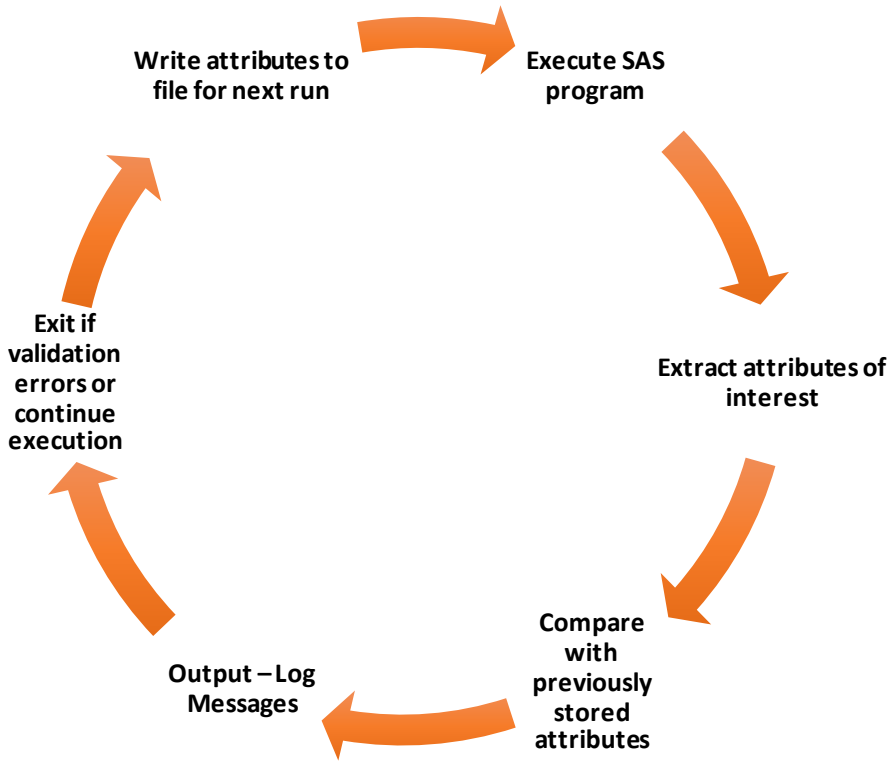
THE TASK

COPY DM DATA FROM STAGING AREA TO ANALYSIS AND REPORTING AREA AND COMPARE KEY DATA ATTRIBUTES

One of the tasks that programmers routinely perform is to copy data from one area to another (for example from data management staging area to analysis and reporting area) and provide feedback to data management if anything is amiss. The examples of issues that a programmer might encounter or interested in tracking is, a seemly new extraction could be the old one (because of data management data extraction having failed for all or some datasets) or missing observations in the dataset in the new extraction compared to previous extraction (incomplete extraction) or missing or new datasets in the current transfer etc. The usual method to keep track of these issues is to store the previous version of the data and run a compare once new data is received. This however, means that one needs to store previous versions of data, which would take resources (especially if the trial is a long-term trial where the size could be huge and therefore prohibitive to some extent) and does not actually serve any purpose. This can also cause version errors (programming using the wrong version of the data instream) if the storage of data is not very systematic.

SOLUTION

We implemented a SAS program that involved extracting data attributes of interest in every execution of the program, reading the stored attributes from previous run to compare with the current, print user messages in SAS logs and if the execution does not hit any validation errors, update the stored attributes from the current execution to serve as memory for the subsequent run. The figure below elucidates this logic.



The first execution step of the macro is the validation step to check if the input and output paths for copying the data are valid, and that the data cutoff date is not a future date, etc. This was followed by reading all the dataset in a macro list and the extraction of attributes of interest from the datasets in the input folder using appropriate SAS proc or function. The below snippet shows the use of 'ATTRN FUNCTION'[1] that returns the value of a numeric attribute for a SAS data set with attrib 'MODTE'[1] that specifies the last date and time that the data set was modified to extract he last modified date for each SAS dataset in the input library.

```
%let dat=%scan(&dlist,&i," ");
%put &dat;
%do %while("&dat" NE "");
     %local dsid rc ;
     %let dsid = %sysfunc(open(&dat));
     %if (&dsid) %then %do;
          %let modte=%sysfunc(attrn(&dsid,modte),datetime.);
          %let rc = %sysfunc(close(&dsid));
          %if &i eq 1 %then %do;
          data &prefix._dttm;
          set &prefix._dttm;
               dataset="&dat";
               modte=input("&modte", datetime20.);
          run;
          %end;
          %else %do;
```

We then compared this information with previously stored information to make sure that at least one dataset has a modified date later than that stored from the last run. If not, then the program is designed to issue a warning stating that the datasets have not changed since the last run and the execution is terminated. If the execution proceeds, the program copies the datasets into the destination folder, updates the data attrib memory file. In addition, it also outputs the names of datasets that weren't modified since the last transfer.

```
%let i=1;
%let moddate=%scan(&datadt,&i," ");
%put &moddate;
%do %while("&moddate" NE "");
    %local rc fileref ;
    %if &i eq 1 %then %do;
        %let parent=&moddate;
        %let rc = %sysfunc(filename(fileref,%qcmpres(&topath/&parent))) ;
            %if %sysfunc(fexist(&fileref)) %then %do;
                %put ERROR: Directory with date &parent already exists, Check if files already copied;
                %goto quit;
            %end;
            %else %do ;
                %sysexec mkdir "%qcmpres(&topath/&parent)" ;
                *-- Over write the the current attrib keeper with a new one ---*;
                filename outfile "%qcmpres(&topath/datmoddt.dat)" ;

                data  _null_;
                    set &prefix._dttm ;
                    file outfile;
                    put dataset $80. modte datetime20.;
                run;

                *-- write a note for all files not modified since last cut ---*;

                %if %sysfunc(exist(&prefix._same_as_prev)) %then %do;

                    filename unmod "%qcmpres(&topath/&parent/not_modified_since_prevcut.dat)";
                    data  _null_;
                        set &prefix._same_as_prev ;
                        file unmod;
                        put dataset $80. modte datetime20.;
                    run;

                    %put NOTE: Some datasets are not modified since last data cut;
                    %put NOTE: To view these check "not_modified_since_prevcut.dat" under the &parent [datacut] folder;
```

Furthermore, the macro also checks for any missing or new datasets in the current transfer that were or were not present in the last copy, and it issues an appropriate warning or user defined notes in the log. Other attributes that are checked are the number of observations in each of the datasets and if any dataset had the number of observation less than the last transfer. The program would issue appropriate warnings in the log (not shown).

The SAS code that we implemented stores information on a flat file (.dat file) on each run to serve as memory for next run, and it will retrieve this information during subsequent executions to print user messages in log, and if the execution was successful, update the memory file for the next run. This is akin to having a memory that stores the requisite knowledge without the need to store previous version of data just for compare purposes.

CONCLUSION

Memory is the function of being able to store information in organized manner and an ability to retrieve the same when needed. The technique presented in this paper is easily extensible to other applications if we can extract the data feature that we are interested in and store the same. However, it's also important to remember that since the interested is not in the artifact itself, but only the features or attributes of interest, its therefore important that all features of interest must we known '**a priori**' as there is no going back to the original artifact.

References

[1] *SAS® 9.4 Functions and CALL Routines: Reference, Fifth Edition*