SouthEast SAS Users Group

THE SESUG INFORMANT

Volume 4, Issue 2

Fall 2002

PRESIDENT'S MESSAGE

F. Joseph Kelley

How time flies. It seems so recently that we were going over the events of SESUG '99 in Mobile, getting ready for SESUG 2K in Charlotte, and making mention of SSU 2001 in New Orleans. Now these great conferences have past and we have SESUG 2002 in Savannah. Once again, I have the privilege of seeing some of the papers we will have, and I am quite pleased as I think you will be. Cochairs Heidi Markowitz and David Maddox have put together a great team of hard working section chairs and the result will be a very solid conference with the usual strong, professional content we expect of a SESUG conference. Plus this will be in the attractive coastal Georgia city of Savannah. As one who was born and grew up there, while I doubt attendees will find quite the image "the book" gave the city, I certainly believe they will enjoy it.

Before any real discussion of SESUG 2002, however, I did want to provide a brief overview of SSU 2001. Co-chairs Debbie Buck and David Riba had so much at that conference: so many papers and events, that it is not possible to detail them. The *Conference Proceedings* alone was the largest we have ever produced; the joint conference itself was 2 ½ days, longer than any that has gone before. This was an outstanding conference and both the South Central SAS Users Group (SCSUG) and the SouthEast SAS Users Group (SESUG) are justly proud of it. Speaking personally, it was a great pleasure to meet and work with the good people of SCSUG. The conference reviews were very positive and yes, we do hope it can be done again. Although the lion's share of the credit goes to Debbie and Dave for the intense effort they put forth, this conference succeeded because of the enormous efforts of

(Continued on page 15)

EDITOR'S MESSAGE

Randy C. Finch

Well, Joe Kelley is back with a vengeance, taking so much space I have to be brief. I would like to welcome Frank DiIorio as our feature writer in this issue. Many of you know Frank from the two books he has authored and the many papers he has presented at conferences. Also, many thanks to John Bentley as he makes his debut in the Informant. We also have articles by our regulars, Ian Whitlock and Kirk Paul Lafler. They have contributed so much to past newsletters that I don't know what I would do without them.

Check out the ads in this issue. Without the financial support of our advertisers, this newsletter would not be free to our readers.

Inside this issue:

The SESUG Informant is a publication of the SouthEast SAS Users Group, Inc. It is published twice a year; in the spring before the SAS Users Group International (SUGI) conference and again in the fall before the SESUG conference.

If you would like to help out by advertising or writing articles for this newsletter, please contact the editor, Randy Finch, by calling 256-386-2197 or E-mailing rcfinch@tva.gov.

Ian Whitlock

Dr. **Ian Whitlock** is a senior systems analyst at Westat working on survey research problems. He has been programming largely in SAS since 1981 using an interactive environment to develop programs for execution in a batch environment. He is active on SAS-L, and has presented many different papers at national, regional, and local SAS User Group meetings.

"A macro provides a service to generate SAS code. The parameters are a contract between the macro (service provider) and the consuming code, i.e. the code that invokes the macro."

GLOBAL MACRO VARIABLES

Let's take a look at the issue of global macro variables. On a number of occasions I have advised that a design goal should be to minimize the use of global variables. Recently (April 2002) on SAS-L, Donald Henderson offered that there are legitimate reasons for global variables so that although one should not abuse global variables, one should not necessarily minimize their use. So what are the legitimate uses?

A macro typically generates SAS code. (The exceptions are macros that consist of purely executable macro instructions such as %PUT, %LET, %IF, and %DO statements without any constant SAS code.) So where does the flexibility to generate different SAS statements on different executions of the macro come from? It comes from the fact that macro variables can hold different values and that macro instructions can do different things based on different values. Can a macro variable be assigned inside the macro so that its value is independent of any outside influence? I suggest not. At first you might think a random number generator might provide an answer. But what seed would you use? If the seed does not change then the values generated do not change. Well zero doesn't change? True, but zero is just an argument to get the seed from the clock time and that is a force outside the macro! You might wish to consider using CALL SYMPUT to convert a data value to a macro variable. But again, I ask, how is the data value created? To be of any use here, it must come from outside the macro. To sum up, macros get their flexibility from having instructions that can do different things based on the different values of macro variables. Those macro variables either exist outside the macro, or they are parameters to the macro assigned by the outside agent invoking the macro, or they are created from information outside the macro.

It should be clear that when a variable is determined by outside information, the program will soon become unreadable if one doesn't convey in the code what outside lines of code are determining the value. Beginners often write macros without parameters and typically do not use tools such as CALL SYMPUT. From the above argument we see that they have to use macro variables defined outside the macro. Typically these variables are global macro variables. This constitutes the primary abuse of global variables. It has two bad effects:

1. The variable is global and therefore known throughout the program. Thus it can easily be changed by mistake and the reader is led into the mistaken thought that the variable is important to the program as a whole, while in fact, it is important only to one macro.
2. The macro becomes difficult to read because there is a reference to a variable and no indication of how that variable got a value. If the variable was not assigned very close to the macro invocation it may be very difficult to locate where the value came from.

The only saving grace to this situation is that macro beginners do not generally write very big or very complex programs. In fact, the technique itself limits the size and complexity of the programs that can be written this way. Thus when a beginner starts down this path, he/she is liable to stay a macro beginner for life.

A macro provides a service to generate SAS code. The parameters are a contract between the macro (service provider) and the consuming code, i.e. the code that invokes the macro. Whenever required information is not contained in the parameter list, either the flexibility or the readability of the macro is harmed. The use of global macro variables to communicate macro parameter information is not a legitimate use. I think Don and I would agree very strongly on this principle.

So what are the legitimate reasons for global macro variables? There is one primary

(Continued on page 4)

Ian Whitlock

"One must use global variables sometimes to cover up defects in the macro language. One common example is the need for static macro variables."

"One of the most common past requests for macro has been to implement the deletion of macro variables. In version 8.2 the command % SYMDEL was added to provide this service."

(Continued from page 2)

reason - the variable controls how the whole program works and thus must be known throughout the program. One common example is a variable to indicate whether the program is running in test mode or production mode. Another comes from wanting to define a number of different, but related files. Thus one might write

```
%let root = C:\Project\my part
                 in project ;
filename progs "&root\progs" ;
libname prj "&root\data" ;
. . .
```

One must use global variables sometimes to cover up defects in the macro language. One common example is the need for static macro variables. A macro MAC must be called a number of times. However the first time some extra initialization code is required. For example we might want a macro to manage PROC APPEND. On the first call we want to wipe out the base data set. After that it is important to not wipe out the base data set. One common solution would be to store a counter that must be known from one execution to the next that counts calls to MAC. Where can the counter be stored? It cannot be inside the macro since all variables defined inside the macro cease to exist when the macro is not executing. Hence the variable must be stored outside the macro. There are two simple possibilities: 1) write a macro to contain all invocations of MAC and make the variable local to this containing macro, 2) put the variable in the global environment. In a sense #1 is begging the question. All global variables can always be avoided by this technique. One has to ask what principle difference is there between a variable known throughout a program and a program with an all-encompassing macro with a variable known throughout this all-encompassing macro. In principle the same problems arise in both cases. The only advantage is when executing interactively the life of the session can be increased by having the all encompassing macro cease to be all encompassing.

One of the most common past requests for macro has been to implement the deletion of macro variables. In version 8.2 the command %SYMDEL was added to provide this service. Thus global no longer means known throughout a program, but only known until deleted. The problem given above about initialization could have been solved with a parameter, say STATE with possible values INIT and CONTINUE. However it is easy to envision a need for a true counter that must be known from one execution to the next. The third solution is to store this information in a data set instead of in a variable. If important enough, this is be best SAS solution, but it means extra time and inconvenience to implement it. As I indicated before, the class of static variables is an important, but missing concept from SAS macro.

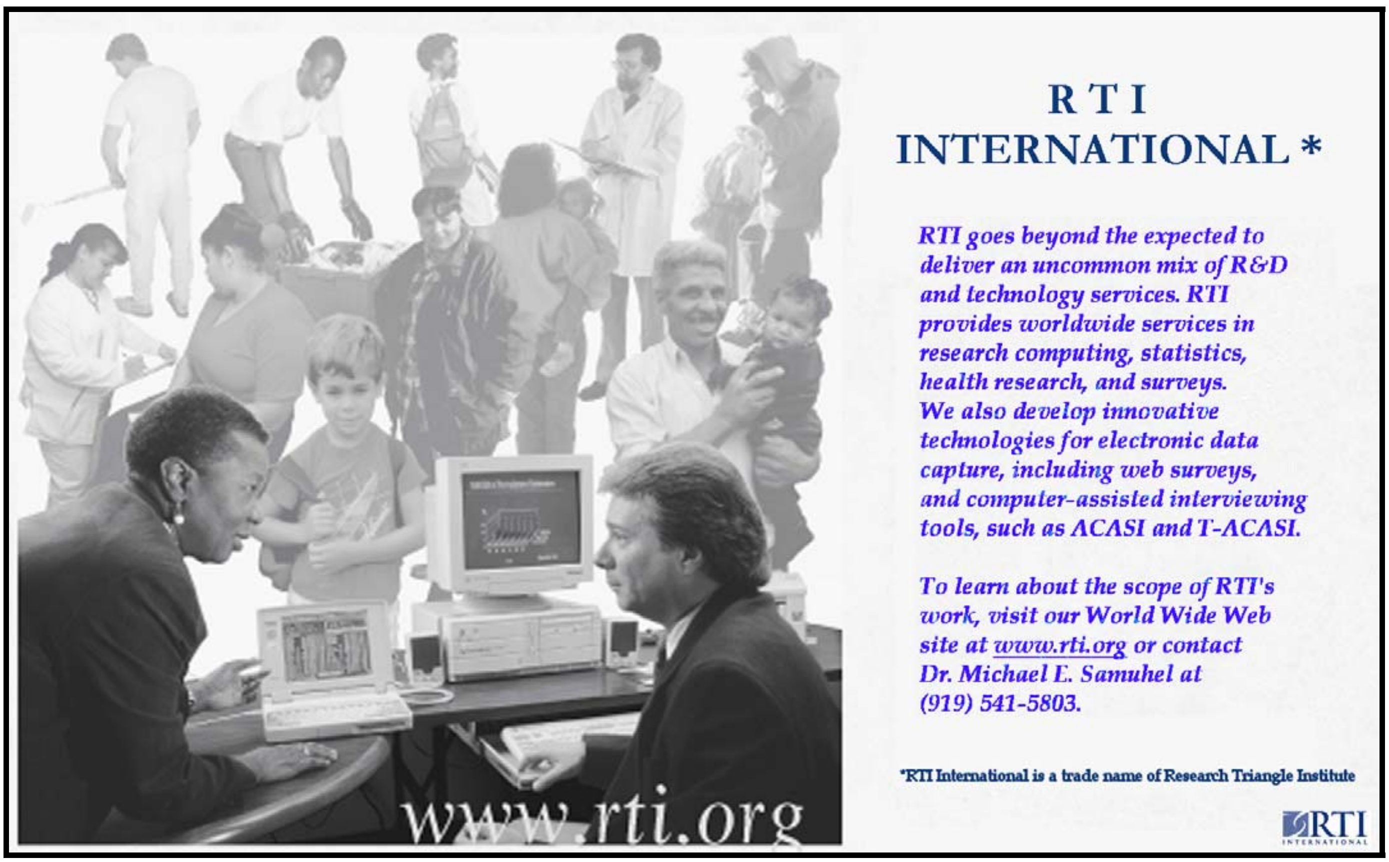
Perhaps Don would see this as a legitimate use of global variables. As I see it, it is an indication that the language is incomplete and I might suffer this use because of language inadequacy, but I would not want to give it a label of legitimacy. It is only a necessary work-around of convenience in common usage.

A similar situation arises when one has several related macros that all must know certain variables. This is a lot like the static situation where it has now become a communication problem between macros instead of a problem of different executions of the same macro communicating. The same solutions occur here as given in the simpler case. Again, it shows a missing construct in the macro language. Object oriented languages have solved this problem by introducing a class as a set of related methods or functions (macros in our case) with a common storage area typically held in the objects that are defined from the class.

Still another possibility is the consideration of program constants. It is often a wise idea to take some literals and define them as constants. For example, the value of Pi, or the size of an array. In some sense these are really program parameters in disguise. I would argue his is not a new legitimate use,

(Continued on page 15)


SMITH HANLEY
Find Your Path With Us
HIGHLY SPECIALIZED PERMANENT & CONTRACT OPPORTUNITIES AVAILABLE
Permanent Placement
Contact: Tracey Gmoser
800.989.5627
Fax: 212.818.9067
perm@smithhanley.com
www.smithhanley.com
• BIOSTATISTICS
• SAS® PROGRAMMING
• DATA MANAGEMENT
• MARKETING SCIENCE
• RESEARCH
• MANUFACTURING
Contract Staffing
Contact: Keith Shelly
800.684.9921
Fax: 407.805.3020
contract@smithhanley.com
www.smithhanley-consulting.com
New York • Chicago • Houston • Southport • Orlando


RTI
INTERNATIONAL *
RTI goes beyond the expected to deliver an uncommon mix of R&D and technology services. RTI provides worldwide services in research computing, statistics, health research, and surveys. We also develop innovative technologies for electronic data capture, including web surveys, and computer-assisted interviewing tools, such as ACASI and T-ACASI.
To learn about the scope of RTI's work, visit our World Wide Web site at www.rti.org or contact Dr. Michael E. Samuhel at (919) 541-5803.
*RTI International is a trade name of Research Triangle Institute
RTI INTERNATIONAL
www.rti.org

Frank DiIorio

Frank DiIorio, currently the entire staff of CodeCrafters, Inc., is author of "SAS Applications Programming: A Gentle Introduction" and (with Ken Hardy) "Quick Start to Data Analysis with SAS." Both titles are part of SAS Institute's Books by Users series and have sold nearly 30,000 copies. Frank has been active in the SouthEast SAS Users Group (SESUG) since its inception, co-chairing the 1994 and 1996 conferences. He has, much to his astonishment after doing the math, over a quarter century experience with SAS software. His new book, "The Elements of SAS Programming Style," (working title) will be published "real soon now." Frank lives in Chapel Hill, North Carolina, the self-professed "Southern Part of Heaven."

VARIABLE CROSS-REFERENCING MACROS - TOOLS FOR WHEN BASE SAS ISN'T ENOUGH

Sometimes, it's just hard to comprehend how robust the SAS tool set is – it seems like every time you have an esoteric need for data analysis or display, there's a function, procedure option, or some other nicety to make your working life productive and easy. All the marketing hype appears to be true.

But then there are the other times. You have a need for something which seems so common, so routine, that you spend an unreasonably long time going through the documentation because what you want just *has* to be there. You eventually discover that yes, indeed, there is no built-in piece of the SAS System that does what you need. Your obvious and routine task will have to be hand-crafted.

This article addresses one of the "other times." It briefly describes the functionality we needed, and then presents one of what are probably many solutions. Examples and some extensions and refinements are also discussed.

The Problem

The impetus for writing the macros that we describe here was straightforward. The client's project required combining data from several studies into a single table. The data were stored in SAS data sets, but their contents varied, usually only slightly: not all variables were in all studies; a variable might have a different length or data type from one study to the next.

What we needed was a visual overview of the studies' contents. We wanted to display the variable lists for each study side-by-side, noting which studies had like-named variables. We also wanted to easily identify discrepancies in data type and/or length.

Let's look at an edited CONTENTS listing for three tables, one per study, and each holding similar data:

study_one

#	Variable	Type	Len	Pos
1	clinic	Char	4	16
4	dob	Num	8	0
5	enroll	Num	8	8
2	patient	Char	5	20
3	race	Num	3	26
6	smoker	Num	3	29
7	treatment	Char	1	25

study_two

#	Variable	Type	Len	Pos
3	age	Num	8	0
9	age_at_onset	Num	8	16
1	clinic	Char	5	28
5	dob	Num	4	24
6	enroll	Num	8	8
2	patient	Char	5	33
4	race	Char	1	38
7	smoke	Num	3	40
8	tmt	Char	1	39

study_three

#	Variable	Type	Len	Pos
3	age	Num	3	24
1	clinic	Char	5	12
5	dob	Num	4	8
6	enroll	Num	8	0
2	patient	Char	5	17
4	race	Char	1	22
7	smoke	Num	3	27
8	tmt	Char	1	23

A visual inspection of these snippets reveals a few problems. CLINIC is character, length 4 in STUDY_ONE, but length 5 in the other tables. AGE_AT_ONSET is present only in STUDY_TWO. These and all others are not too hard to locate when there are few variables involved. Consider, however, the potential for tedium and error if many tables and/or variables have to be compared. Clearly, there is a need for automation.

The Approach

The initial design of the cross-reference utility was simple. Read the dictionary tables – SAS's meta data – and extract information about two or more data sets. Massage the data so that each variable in the "report" data set represents a table, and

(Continued on page 8)

PRICEWATERHOUSECOOPERS

PricewaterhouseCoopers LLP is the world's largest professional services organization committed to helping clients build value, manage risk, and improve their performance. PwC Consulting, a Business of PricewaterhouseCoopers LLP, solves complex business problems for top-tier organizations, such as postal administrations worldwide. We draw on our expertise in the postal industry to provide measurable change to help our clients improve and succeed.

Our Springfield, VA office is currently looking for a Statistician/Analyst to join our dynamic team.

Qualified candidates will possess the following skills:

- 2 – 3 years SAS programming
- Experience using techniques such as macros and arrays to transform and analyze complex data
- Ability to synthesize complex data for analysis and presentation to clients
- Solid communication and presentation skills
- Demonstrated initiative
- Experience leading small teams or projects
- Ability to work in a fast-paced, deadline-oriented environment
- BS or MS in a quantitative, analytical field, such as mathematics, statistics, operations research, psychology, or economics

Qualified candidates please submit your resume and salary requirements to: angela.martenson@us.pwcglobal.com

LOCAL CANDIDATES ONLY. NO AGENCIES OR THIRD PARTY PHONE CALLS PLEASE.

PwC Consulting is proud to be an Affirmative Action and Equal Opportunity Employer.

Frank Dilorio

"You eventually discover that yes, indeed, there is no built-in piece of the SAS System that does what you need. Your obvious and routine task will have to be handcrafted."

"Now we add a small wrinkle to the design – how do we identify type and length mismatches? Since this information is also in the meta data, extraction doesn't present a problem. Presentation is tricky, though.“

(Continued from page 6)

each observation represents a variable. If a particular study-variable combination exists in the meta data, the value in the report data set becomes "[ok]". If not, it is blank. So, if the studies meshed perfectly, the report data set would be a matrix filled with "[ok]".

Now we add a small wrinkle to the design – how do we identify type and length mismatches? Since this information is also in the meta data, extraction doesn't present a problem. Presentation is tricky, though. We want to report *mismatches*, but also don't want to overload the reader with cell after cell of data for *matches*. Essentially, we wanted to highlight the bad news and report the good news sparingly. The decision was to create variable TYPE_LEN, containing type and length. If the types and lengths for all the data sets matched, TYPE_LEN would have that information and the corresponding data set fields would be set to "[ok]". If we had a mismatch between two or more datasets, TYPE_LEN would be blank, and the data set fields' individual values would contain information about type and length.

Another twist that presented itself early on was what, exactly, to report from the report data set. In some situations, a listing of all variables might be helpful. In other cases, only mismatches are of interest. Other views of the data would be required as the project progressed. To provide this flexibility, a series of 0/1 indicator variables was attached to the data. These are used as filters for the REPORT procedure code that actually does the display.

The Code

The program was divided into two macros. The first, CROSS_REF_SETUP, parses the list of data sets to compare, makes sure they exist, then builds an analysis data set for each of them. The code is shown below (with some stylistic concessions to accommodate the margins):

```
%macro cross_ref_setup / parmbuff;
%global n_data;
%let line   = &syspbuff.;
%let length = %length(&line.);
%let line   = %qsubstr(&line., 2,
                       %eval(&length.-2));
%let n_data = 0;
%do num = 1 %to 99;
  %let piece = %scan(&line., &num.,
                     %str( ) );
  %if &piece. ^= %then %do;
    %if %index(&piece., .) = 0 %then
      %let piece = work.&piece.;
    %if %sysfunc(exist(&piece.))
                     %then %do;
      %let n_data = %eval
                    (&n_data. + 1);
      %let libname&n_data. =
           %upcase(%scan
                   (&piece., 1, .)) ;
      %let memname&n_data. =
           %upcase(%scan
                  (&piece., 2, .)) ;
      %end;
    %end;
    %else %if &piece. = %then
          %let num = 100;
%end;
%if &n_data. = 0 %then %goto bottom;

proc sql;
  %do i = 1 %to &n_data.;
    %put Data set &i. of &n_data:
         &&libname&i...&&memname&i;
    create table _temp&i._ as
    select libname,
           memname,
           name,
           trim(type) ||
               ' ' ||
           left(put(length, 5.)) as
                           specs&i.
    from dictionary.columns
    where   libname = "&&libname&i."
          & memname = "&&memname&i."
    order by name;
  %end;
quit;

data report;
  merge %do i = 1 %to &n_data.;
          _temp&i._(in=_&i)
        %end;
        ;
  by name;
  length compare $10;
  array specs(&n_data.) $10;
  status       = 1;
  done_compare = 0;
  do i = 1 to dim(specs);
    if specs(i) ^= ' ' &
                ^done_compare
      then do;
      compare      = specs(i);
      done_compare = 1;
      end;
    if specs(i) ^= ' ' &
       specs(i) ^= compare then do;
       status = 0;
       leave;
       end;
   end;
   if status = 1 then do;
     do i = 1 to dim(specs);
       if specs(i) ^= ' ' then do;
         type_len = specs(i);
```

(Continued on page 12)

Join your friends and colleagues at the

Tenth Annual SESUG Conference

September 22-24, 2002
Savannah, GA

Conference Chairs:

Heidi Markovitz
Simply Systems
HeidiM@Simply-Systems.com

David Maddox
BellSouth Communications, Inc.
David.Maddox@BellSouth.com

www.sesug.org/SESUG02

Come to historic Savannah, the "Hostess City" to learn what's classic and what's new in SAS software. This year's conference may be the biggest training bargain you'll see. For around $300 (see details on Registration Web page) you will get

Training and Inspiration

- ⇒ Over 100 papers and posters in 10 technical sections
  - o including 6 Hands-On workshops taught by Destiny, Leaders in SAS training.
- ⇒ Large SAS Demo Area staffed by experts in SAS products and techniques from SAS Institute.
- ⇒ Demonstrations and displays by popular vendors such as Destiny, FullSeven Technologies, and Network Appliance.
- ⇒ Opening keynote by Paul Kent, Director, Base SAS Research, SAS Institute on "SAS: Still Relevant after All These Years".
- ⇒ Closing Keynote by C. Olivia Parr Rud, Founder and CEO, Olivia Group on "The 4 C's of SuCCCCess: Creativity, Communication, Cooperation and Change".

Food and Drink

- ⇒ Sunday night Welcome Reception
- ⇒ Monday "after class" Demo Area Mixer, hosted by SAS Institute
- ⇒ Monday night get together at a Savannah night spot with live local entertainment
- ⇒ Continental breakfast Monday and Tuesday morning, and continuous beverage service during the training day.

Freebies and Discounts

- ⇒ On-site discounts on all SAS Institute publications.
- ⇒ Eligibility to participate in drawings for valuable prizes from SAS Institute and other vendors (TBA).
- ⇒ SESUG and vendor souvenirs - pens, t-shirts…
- ⇒ Copy of the conference Proceedings, a most valuable reference to use at work.

Extras for your convenience and enjoyment

- **16 Weekend Workshops** Optional mini-courses available at the conference hotel on Friday, Saturday, Sunday, and Tuesday afternoon. These classes are taught by SAS professionals and experienced trainers. At $125 for a half-day and $250 for a full-day (extra-fee for non-conference attendees) they are an economical way to pump up your conference experience.
- **Guest Program** Bring along your spouse, buddy, or kids. For $50 per person your guests can join you at the Opening session and all mixers or social events. They receive a full-complement of drink tickets, SESUG souvenirs, and their choice of Monday activity. They can attend either a 1-hour narrated Savannah River sightseeing cruise or a walking tour of the Savannah Historic District.
- **Sunday Historic Savannah City Tour** Get acquainted with Savannah and your fellow conference-goers on this optional two-hour narrated trolley tour. Relax and enjoy the surroundings before getting down to work— also perfect for those attendees who are traveling with family. Tickets for this optional activity are $20.

Details

Dates: September 22-24, 2002

Location: Hyatt Regency Savannah – Two West Bay Street – Savannah, GA 31401 – 912-238-1234

Schedule: Registration Desk opens Sunday at Noon
Opening session is Sunday at 7:30 PM
Academic sessions start at 8AM and go until 5PM
Closing session is expected to end Tuesday by 2PM

Prices: $330.00 Single conference registration, postmarked by September 12, 2002
$375.00 On-site conference registration (after September 12)
$125/$250 Weekend Workshop registration ($25/per workshop additional for non-conference attendees)
$50 Guest Program
$20 Pre-conference Savannah Historic District Trolley Tour

Hotels: Hyatt Regency Savannah – Two West Bay Street – Savannah, GA 31401 – 912-238-1234 $135/night (single/double)
Days Inn & Suites - 201 W. Bay Street (one block from the Hyatt) – 912-236-4440 - Fax 912-232-2725 - $89/night (single/double) or $109 (one-bedroom suite)
NOTE: Call the hotel and ask for the SESUG rate (guaranteed until 8/28 or our room block fills up). Hotel rates do not include parking and taxes.

Web Site: www.sesug.org/SESUG02

Registration: Register on-line or by FAX or mail. Go to www.sesug.org/SESUG02, click on Registration. Either download the Registration Form or click on the Online Registration.

Payment: SESUG accepts payment by check or credit card (AmEx, VISA, Mastercard, Discover).

Questions? For questions regarding Registration contact our Off-Site Registrar Debbie Skinner by email at deborah.skinner@us.fortis.com or phone or Fax at (916) 353-2992. Her mailing address is 238 Briggs Ranch Dr., Folsom, CA 95630.

Frank DiIorio

"It's not hard to think of extensions to the macros beyond adding new values to the TYPE parameter. Macro variable output is a likely candidate."

(Continued from page 8)

```
         specs(i) = '[ok]';
         end;
     end;
     end;

   if sum(of _1-_&n_data.) = &n_data.
                     then do;
     inall = 1;
     notinall = 0;
     end;
     else do;
       inall = 0;
       notinall = 1;
       end;
   if type_len = ' ' then
                     mismatch = 1;
      else           mismatch = 0;
   if ^mismatch & inall then
                     matchall = 1;
      else           matchall = 0;
   if ^mismatch then matchfound = 1;
      else          matchfound = 0;

   label %do i = 1 %to &n_data.;
              specs&i.="&&libname&I
                         &&memname&i"
         %end;
         type_len = "Specs"
         ;
run;

%bottom: ;
%mend;
```

The macro accepts blank-delimited data set names as its parameter. We use the EXIST function to test for the presence of the data set. If found, we increment N_DATA and create macro variables LIBNAMEn and MEMNAMEn. We are, in effect, building two arrays, each N_DATA elements in length.

The macro then constructs N_DATA data sets using CREATE TABLE statements in SQL. Data sets _TABLE1_ to _TABLE_n contain library, member, and variable names, and TYPE_LEN, which looks like "char 40", "num 8", and so on. The last piece of the process is the DATA step creating data set REPORT. Here we combine the tables and create the indicator variables used for filtering. REPORT can now be used as input to display, or other, procedures.

The second macro in the program, CROSS_REF_USE, accepts an optional filtering parameter and writes an HTML file containing a report.

```
%macro cross_ref_use(type=);
%put TYPE: &type.;
%let type = %upcase(&type.);
%if &type. = MATCHFOUND %then %do;
  %let title2 = Matching Whenever
                                Found;
  %end;
  %else %if &type. = MATCHALL
                          %then %do;
    %let title2 = Matching in Every
                       Data Set;
    %end;
  %else %if &type. = INALL %then %do;
    %let title2 = In Every Data Set;
    %end;
  %else %if &type. = MISMATCH
                          %then %do;
    %let title2 = Mismatches Only;
    %end;
  %else %if &type. = NOTINALL
                          %then %do;
    %let title2 = Missing From At
                  Least One Data Set;
    %end;
  %else %let title2 = All Variables
                      Were Selected;

proc report headline nowindows
     data=report %if &type. ^= %then
         (where=(&type.=1)) ;
     split=' ';
columns name type_len
        %do i = 1 %to &n_data.;
            specs&i.
        %end;
        ;
%do i = 1 %to &n_data.;
   define specs&i. / display center ;
%end;
title "Variable Comparisons Taken at
  &systime. on &sysday., &sysdate.";
%if &title2. ^= %then %do;
    title2 "&title2.";
    %end;
run;

%mend;
```

The process is straightforward. Assign TITLE2 on the basis of TYPE, then filter REPORT, created by macro CROSS_REF_SETUP, and create the report. The general-purpose nature of REPORT allows multiple calls to CROSS_REF_USE with only one call to CROSS_REF_SETUP. Valid report types (parameter TYPE values) are:

- MATCHFOUND Restrict the listing to variables with matching characteristics in at least one of the data sets. Contrast with MATCHALL, below.
- MATCHALL Restrict the listing to variables with matching characteristics in each of the data sets. Contrast with MATCHFOUND, above.
- INALL Restrict the listing to variables

(Continued on page 14)

SESUG '03

St. Pete Beach, FL

September 21-23, 2003

Conference co-chairs:

Steve Noga
snoga@rhoworld.com

Gary Schlegelmilch
sesug03@juno.com

Join us as we go "Back to the Beach" for the 11th annual conference of the South-East SAS Users Group! Our first conference was held waaaaay back in 1993, at the beautiful Trade Winds Island Resort in St. Pete Beach, Florida. We've traveled the range from Norfolk to New Orleans since then; but to kick off our second decade, we wanted to go back to where we began. All in all, we're planning a fabulous conference; one where you'll be able to share ideas with SAS people from all over the region, get an incredible wealth of information for a fairly small investment of time, and have it all happen while you're having a great time on the lovely St. Pete Beach.

Hope to See You in St. Pete!

Frank DiIorio

(Continued from page 12)

found in every data set, regardless of characteristic matching status.

- MISMATCH Restrict the listing to variables with type and/or length mismatches in two or more data sets.
- NOTINALL Restrict the listing to variables that are not found in every data set, regardless of characteristic matching status.

If TYPE is not assigned, it defaults to a null value. This creates a listing of all variables in the data sets.

Examples

Suppose we want to list all variables

```
ods html body= "C:\papers\var_xref
                    \var_xref.htm";
%cross_ref_use
ods html close;
```

Output, using our sample data, is found in Figure 1. Variables are displayed in alphabetical order, one variable per row. A blank SPECS value signals type/length conflicts between two or more data sets (AGE, CLINIC, DOB, and RACE have this problem). The presence of a matching variable in a data set is indicated by "[ok]", while present, but conflicting variables, have the type and length for that variable in the data sets.

To create a report containing variables that are found in every data set, we enter the following code. Output is found in Figure 2.

```
ods html body= "C:\papers\var_xref
                    \var_xref.htm";
%cross_ref_use(type=matchfound)
ods html close;
```

Finally, to list nothing but mismatches data types and lengths, we enter the following. Output is found in Figure 3.

```
ods html body= "C:\papers\var_xref
                    \var_xref.htm";
%cross_ref_use(type=mismatch)
ods html close;
```

Extensions

It's not hard to think of extensions to the macros beyond adding new values to the TYPE parameter. Macro variable output is a likely candidate. For example, a program may need to process only variables with matching characteristics in all studies. In this situation, a static report such as the one generated by CROSS_REF_USE isn't helpful. What can be done is use the REPORT data set created by CROSS_REF_SETUP and cre-

(Continued on page 15)

Figure 1: Using Default Value of TYPE

Variable Comparisons Taken at 09:49 on Thursday, 07FEB02
All Variables Were Selected

Column Name	Specs	WORK STUDY_ONE	WORK STUDY_TWO	WORK STUDY_THREE
age			num 3	num 8
age_at_onset	num 8			[ok]
clinic		char 4	char 5	char 5
dob		num 8	num 4	num 4
enroll	num 8	[ok]	[ok]	[ok]
patient	char 5	[ok]	[ok]	[ok]
race		num 3	char 1	char 1
smoke	num 3		[ok]	[ok]
smoker	num 3	[ok]		
tmt	char 1		[ok]	[ok]
treatment	char 1	[ok]		

Figure 2: Variables Found in Every Data Set

Variable Comparisons Taken at 09:49 on Thursday, 07FEB02
In Every Data Set

Column Name	Specs	WORK STUDY_ONE	WORK STUDY_TWO	WORK STUDY_THREE
clinic		char 4	char 5	char 5
dob		num 8	num 4	num 4
enroll	num 8	[ok]	[ok]	[ok]
patient	char 5	[ok]	[ok]	[ok]
race		num 3	char 1	char 1

Figure 3: Display Only Mismatches

Variable Comparisons Taken at 09:49 on Thursday, 07FEB02
Mismatches Only

Column Name	Specs	WORK STUDY_ONE	WORK STUDY_TWO	WORK STUDY_THREE
age			num 3	num 8
clinic		char 4	char 5	char 5
dob		num 8	num 4	num 4
race		num 3	char 1	char 1

Frank DiIorio

(Continued from page 14)

ate a list of variables for use in, say, a KEEP data set option.

```
proc sql;
select name into :_matchall separated
                              by ` `
from report
where matchall = 1;
quit;

data combined;
set study_one(keep=&_matchall.)
    study_two(keep=&_matchall.)
    study_three(keep=&_matchall.);
run;
```

Other modifications and extensions will be made as new demands are placed on the application. It's likely that many such changes would simply be different uses of the REPORT data set.

Contact

If you have comments, contact me at FCD1@MINDSPRING.COM. You're welcome to use the macros, so long as attribution is made.

F. Joseph Kelly

(Continued from page 1)

so many equally hard working, dedicated volunteers. It was great!

SSU 2001 set a high standard, but the quality of SESUG 2002 should be a match. There will be a number of papers in a new section, "Data Management", that will explore areas of data warehousing, data cleaning and data manipulation. As at SSU, the former "Basic" and "Advanced" Tutorials are combined in a single robust section and will have presentations in many areas and focusing on many skill levels. These are always popular, and everyone should find something they will not want to miss. Other sections are similarly strong and attendees will want to review the final list of presentations with some care: with so much to choose from, it can be difficult to make a choice. I look forward to seeing all of you in Savannah.

And after Savannah? Well, in 2003 SESUG will be in the Tampa Bay area. Our Co-chairs will be Steve Noga of Rho, Inc. and Gary Schlegelmilch of the Census Bureau. Steve and Gary have been at work on this for a few months already, but they will move to the fore at the conclusion SESUG 2002. They have already shown both enthusiasm and thoughtfulness and I am already looking forward to SESUG 2003.

Those who have followed this column will know that I always like to add my own "call for participation":

The SAS User Group conferences, whether at a regional level, such as SESUG, or at a national level, such as SUGI, depend upon the work of many volunteers. The material recompense for this is small; yet these volunteers are essential to the Conference. I thank all of them. Still, by their involvement, no matter the level, the volunteers have enhanced their own conference experience. I urge all who can to sign up. The work of a Session Coordinator or Registrar's Assistant is not difficult, but you will see and participate in the conference in a new way. Speaking as one who started one conference by helping to assemble the poster stands, I assure you that you will enjoy it. And heck, one day you may even find yourself chairing one.

What? There's still room? My good friend (and secret enemy), the editor, has not appropriated it for his own uses? Well then, I wanted to say that [segment deleted, -ed.]

Ian Whitlock

(Continued from page 4)

but rather a further explication of the program parameters that I presented first.

We have not really seen any new SAS features (shiny language toys) in this discussion, but hopefully it has helped you to see some of the issues in writing and maintaining large SAS programs when using SAS macro tools. Even if it does not change the way you write programs, I would hope that it makes you think more about the programs that you do write.

John Bentley

Using SAS to Extract From a Parallel RDBMS

Using SAS to pull data from a relational database like Oracle, DB2, or Informix has always been common, and the RDBMS libname engines make it even easier. But it isn't always the most efficient way to do it, especially working in a client-server environment with a large database. Here we will look at the real-time performance of a series of four tests that use different approaches to extract data from a remote relational database and land a SAS data set on a desktop computer.

In a client-server environment, processing required to access and manipulate the raw data can be distributed across the platforms in a number of ways. SAS provides two services that allow distributed processing: Remote compute services allow utilization of the computer resources of the remote system, that is, you can run jobs there. Remote data services allows access to data, such as SAS data sets and catalogs, flat files, and external databases, stored on a remote platform. Remote data services is subdivided into data transfer services, by which a data set or file is copied from one platform to another, and remote library services (RLS), which provides access to remote data as if it were stored locally without creating a local copy.

In these tests, we're creating a SAS data set from a terabyte class Informix data mart on a massively parallel IBM system. Two tables are accessed: one has 3.2 million records; the other has 6.6 million records. SAS is installed on one 4-way node. SAS Version 8.2 is on both the SP and a Dell Optiplex desktop. SAS/CONNECT establishes a remote session on the SAS node from the desktop and SAS/Access to Informix allows us to extract data from the database.

First Test—Remote Library Services

In the first test we use remote library services to extract data and land it on our desktop. By using the Informix engine in our remote librefs and then mapping that librefs for local access, we can treat the tables in the Informix database as local SAS data sets and use either PROC SQL or a DATA step to extract the data. Here's the way we set up the librefs.

```
rsubmit;
  libname risk informix
    database="rmad@sigmasoc.1"
    read_isolation_level=dirty_read;
endrsubmit;

libref risk remote server=productn;
libname locExt 'c:\temp';
<proc sql or a data step to extract
the data>
```

In the log, we see that our extract data set is created with 23164 rows and 6 columns and that the extract took 22:28.80 of real time.

Second Test—Remote Compute Services and Data Transfer Services

Next, we use a combination of remote compute services and data transfer services. We still use the Informix engine in our remote libref, but we run the extract code on the remote platform instead of the locally and we use PROC DOWNLOAD to move the data to our desktop.

```
libname locExt 'c:\temp';

rsubmit;
  libname risk informix
    database="rmad@sigmasoc.1"
    read_isolation_level=dirty_read;
  libname remExt '/work/tmp/d507201';

  <proc sql or a data step to extract
   the data>
  <proc download>
endrsubmit;
```

Reviewing the log, we see that the extract data set still has 23164 rows and 6 columns but the total real time was 14:52.01, almost seven minutes faster than the first test.

Third Test—SQL Pass-Through Facility and Data Transfer Services

Now, instead of using the Informix engine with our remote LIBNAME statement we use the SQL Pass-Through Facility. SAS doesn't generate the SQL but instead passes

(Continued on page 17)

Since 1987 **John Bentley** has used SAS Software in the healthcare, insurance, and banking industries. He is currently an Assistant Vice-President with the Corporate Data Management Group at Wachovia Bank with responsibilities for the development of SAS client-server applications to extract, manipulate, and present data from data warehouses and data marts. John is a SAS Certified Professional V8 and regularly presents at SAS User Group Conferences. He is chairing the Systems Architecture section at SUGI 28 next year.

Contact Information
John E. Bentley
Wachovia Bank
2301 S. College Street
NC-1025
Charlotte, NC 28288
704-383-2686
John.Bentley@wachovia.com

(Continued from page 16)

the SQL we specify directly to Informix.

```
libname locExt 'c:\temp';

rsubmit;
  libname remExt '/u/d507201';
  proc sql;
    connect to informix
        (database="rmad@sigmasoc.1");
    execute (set isolation to
             dirty read) by informix;
    <query to create a SAS data set>
    disconnect from informix;
  quit;
  <proc download>
endrsubmit;
```

In this test, the extract data set still has 23164 rows and 6 columns but took only 01:34.63 to be created and moved to the desktop. Quite a difference!

Fourth Test—SQL Pass-Through Facility using an Informix Stored Procedure and Data Transfer Services

When working with Informix, SAS's Pass-Through Facility supports an EXECUTE statement that sends stored procedures, SQL statements, and even entire queries directly to Informix for execution. The syntax is:

```
execute (<procedure, SQL, or query>)
         by informix;
```

In our data mart environment, we use this EXECUTE statement a lot. When extracting from or joining large tables, performance is improved by first extracting from the tables into a memory table and then writing that result set to a SAS data set. With <u>very</u> large tables or many joins, we've found that it's best to extract the data into a series of memory tables, manipulate and join those memory tables into other memory tables, and then write the final results set to a SAS data set.

Here's the code that was tested.

```
rsubmit;
  libname remExt '/u/d507201';

  proc sql;
    connect to informix
        (database="rmad@sigmasoc.1");
    execute (set isolation to
             dirty read) by informix;
    execute (
      select <fields>
      from <two tables>
      where <conditions>
      into scratch temp_cs
    ) by informix;

    create table remExt.cs as
    select * from connection to
                        informix
         (select * from temp_cs);

    disconnect from informix;
  quit;

  <proc download>
endrsubmit;
```

This query ran in 49.87 seconds, in line with what we expected.

Discussion and Conclusions

What conclusions can we draw from the results of these four tests?

First, Remote Library Services is not efficient *when working with large data sets*. When we consider how Remote Library Services works, this is no surprise. RLS brings the data across the network one record at a time as the local application requests it. In our test, we have the additional overhead of first pulling the data from a parallel database spread across multiple nodes. Our source data tables are moderately large, and each record had be extracted and brought across the network at least once to satisfy our query's Where clause. Since the data is not landed on desktop in a temporary data set, if the same record is requested again it must be moved again

Second, the performance of the Remote Compute Services and Data Transfer Services test was a bit surprising. More investigation is warranted, but the most critical performance inhibitor may be that because the data are partitioned across multiple database nodes SAS is bringing millions of records across the system's internal network to the SAS node for evaluation by the Where clause. A second reason could be that the SQL being generated by SAS might not be the most efficient. We'd have to look at it before making a judgment about that, though. (To see the SQL SAS is passing to the RDBMS, we'd use OPTION SASTRACE=',,,d';)

Finally, using the SQL Pass-Through Facility and Data Transfer Services is quite clearly the best performer and the reason is clear. SAS passes the SQL query to the data-

(Continued on page 18)

John Bentley

"Here we will look at the real-time performance of a series of four tests that use different approaches to extract data from a remote relational database and land a SAS data set on a desktop computer."

Kirk Paul Lafler

Kirk Paul Lafler is a SAS Alliance Partner® and SAS Certified Professional® with over 25 years of SAS software experience. He provides hands-on SAS training around the world. Kirk has written over one hundred articles for professional journals and SAS User Group proceedings. His popular SAS Tips column appears regularly in the SANDS and SESUG Newsletters. His expertise includes application design and development, training, and programming using base-SAS, SQL, ODS, SAS/FSP, SAS/AF, SCL, FRAME, and SAS/EIS software.

SORTING TIPS

It's been said by many a SAS user that the SORT procedure is one of the easiest procedures to use and one of the most abused. In this issue, I'd like to share a strategy I've used with some success when processing, and in particular sorting, large data sets. You may have even used one or more parts of this strategy on your own data sets. The best approach any user can follow when sorting large data sets is to first spend time understanding what your objective is with the sort, and then spend a little more time planning how to best achieve that objective.

Avoiding problems related to sorting

The trick to performing a successful sort is to first understand what your objective is, and then find out how to achieve that objective. (Sometimes a little luck never hurts either.) Here are a few guidelines to follow whenever you find yourself sorting a large data set.

1. Determine if the data in a large data set is already in sorted order. If it is, then a sort may not be necessary.
2. Determine whether the data is in a particular sort order by specifying the BY statement in a DATA or PROC step (other than the SORT procedure). If the data isn't in the desired sort order specified in the BY statement, a runtime error is produced.
3. Specify the NOTSORTED option when the observations with the same BY value are grouped, but the observations within the group are in unsorted order.
4. Determine whether any installation or resource constraints (e.g., shortage of space, etc.) exist that could prevent a data set from being sorted.
5. Understand the requirements of the DATA or PROC step that you plan to use as he input data set. Many procedures, including all of the summary procedures, don't need data in sorted order because they produce results in sorted order after summaries are produced anyway.
6. Consider using a tag sort if a sort is needed. When a data set is too large to fit in memory, then tag sorting may be more efficient because it requires less temporary storage space and memory. Specify the TAGSORT option in the SORT procedure statement.
7. Consider only sorting the observations and variables that are needed to satisfy the conditions of the analysis.

If you would like more information or have any questions about these tips, please contact: Kirk Lafler, Software Intelligence Corporation at the following e-mail address KirkLafler@cs.com.

John Bentley

(Continued from page 17)

base where a copy of it runs simultaneously on each of the database nodes. If the data are partitioned onto 10 database nodes, for example, then we will have ten queries running at once, each against 1/10th of the total data. Only the results sets are returned to the SAS node to create the final data set.

Using SQL Pass-Through with the EXECUTE statement turned in the best performance, and we could have perhaps shaved a few more seconds off the execution time by extracting the fields we need from each of the source data tables to it's own memory table and then joining those memory tables to get the final result set. If we were working with tens of millions of records then we definitely would have used this multiple memory tables approach.

In summary, we can say that because SAS provides more than one way to extract from an RDBMS in a client-server environment it's important to (1) understand the strengths and weaknesses of each of the extract approaches that SAS allows; (2) understand the configuration of your particular system and database; (3) have an idea of the how much data you will be in extracting and landing your final data set; and (4) for a production application, testing the different approaches and ways of specifying the query may be time well spent.

Save 20% on SAS books while at the conference.*

Visit the SAS® Publishing Booth to learn more about...

- **New Books**
- **Books by Users℠ Press** - users wrote them, users love them
- **SAS Online Tutor™**- web-based training contains 35 lessons that teach how to use Version 8 SAS environment
- **SAS® Certified Training: Core Concepts, V8** - contains instruction designed to prepare you for the SAS Certified Professional V8 Exam (A00-801)

SAS Publishing offers you a comprehensive library of documentation for each component of the SAS Software System. Stop by the booth, place your order and receive a **20% discount** on your SAS documentation purchase. SAS staff will be on hand to answer any questions and to assist you with your order.

Enjoy the conference!

SAS Publishing

The Power to Know™

SouthEast SAS
Users Group

Index to Advertisers

We're on the Web!
www.sesug.org

SouthEast SAS Users Group

SESUG EXECUTIVE COUNCIL MEMBERS	TELEPHONE	E-MAIL
Gregory Barnes Nelson (1999 co-chair)	919-465-0322 x351	greg@thotwave.com
Dan Bruns, Vice President (1997 co-chair)	423-751-6430	debruns@tva.gov
Deborah Buck (2001 co-chair)	281-256-1619	dbbuck@compuserve.com
Philip J. d'Almada, Secretary (2000 co-chair)	404-639-6120	pdalmada@cdc.gov
Frank C. DiIorio (1994 & 1996 co-chair)	919-942-2028	fcd1@mindspring.com
Randy C. Finch, Treasurer, Editor (1998 co-chair)	256-386-2197	rcfinch@tva.gov
Maribeth H. Johnson (2000 co-chair)	706-721-3785	maribeth@stat.mcg.edu
F. Joseph Kelley, President (1998 co-chair)	706-542-5359	jkelley@arches.uga.edu
Andrew T. Kuligowski (1997 co-chair)	727-738-3000 x5203	kuligoat@tvratings.com
David Maddox (2002 co-chair)	205-977-0415	david.maddox@bellsouth.com
Heidi Markovitz (2002 co-chair)	305-365-0439	HeidiM@Simply-Systems.com
George Matthews (1999 co-chair)	706-542-5359	gmatthew@arches.uga.edu
Steve Noga (2003 co-chair)	919-408-8000 x235	snoga@rhoworld.com
S. David Riba (1993 & 2001 co-chair)	727-726-6099	dave@jadetek.com
Gary E. Schlegelmilch (2003 co-chair)	703-670-0451	geschlegel@juno.com

The SESUG Informant
POB 8345
Clearwater, FL 33758