

Improving Efficiency using the "Virtual Merge"

Steve Sanders, Regions Financial, Birmingham, AL

ABSTRACT

Merging data sets is a critical part of relational database processing, but the required sorting of data is one of the most significant contributors to total processing time. This paper will demonstrate a method to use the macro processor to simultaneously perform multiple lookup table merges and eliminate large amounts of processing time by eliminating the need to actually sort and merge the master data set with each lookup table.

INTRODUCTION

When using enterprise relational database systems, the need to merge very large tables with a number of very small lookup tables is a quite common, and extremely time consuming task. When there are several lookup table merges required, the time to repeatedly sort the master (destination) table can quickly cause a program to run so long that it loses value.

Using Proc Format to create a custom format from a table is one way to eliminate merging tables. The "virtual merge" technique described in this paper performs a similar function using string concatenation along with the symput function to tokenize lookup table values into macro variable symbols. These symbols are subsequently resolved using the symget function to create a variable in the destination table that contains the value from the lookup table. As an added benefit, you can put this entire process into a macro to utilize a local symbol table so all of the storage space used by the symbols will be automatically released when the macro completes. Although the example in this paper only replicates a simple merge of two datasets, the concept can be used to perform multiple "merges" at one time with any number of matching keys.

The code examples used in this paper utilize data available in the SASHelp library. Therefore, the code will run in any SAS® session.

THE "VIRTUAL MERGE", STEP BY STEP (THERE'S ONLY TWO)

For demonstration purposes, we will capture variable metadata from SASHelp.VColumn and then append the Number of Variables, Number of Observations, and Modification Date of the table containing each variable from SASHelp.VTable.

Step One: Tokenize the values in the lookup table(s) that are to be "merged" with the master data set.

This is accomplished using a null data step to create a series of distinct macrovariable symbols. The names of these symbols are created by concatenating key data from the lookup table to a string that will make the multiple sets of symbols distinct. The symbols are created containing the values from the variable(s) to be "merged" from the lookup table.

```
/* This creates two sets of macrovariables. The first set contains the
   Modification Date and the second set contains the result of concatenating
   the Number of Variables and the Number of Observations in each table. */
data _null_;
  set SASHelp.VTable;
  where Libname="SASHELP";
  MVarName1 = catt("Mod_Date_",LibName,"_",MemName);
  call symput(MVarName1,compress(put(moddate,best.)));
  MVarName2 = catt("NObs_Data_",LibName,"_",MemName);
  call symput(MVarName2,compress(catt(put(NVar,best.),"~",put(NObs,best.))));
run;
```

To demonstrate how to capture one or more variables from the lookup table, two sets of symbols were created in the null data step. The first set contains just one variable value (Modified Date), and the second contains two variables (NVar and NObs) that have been concatenated together with a tilde (~) as a delimiter. It is possible to concatenate as many variables as is needed into one set of symbols. The only limitation is that the maximum length of a macrovariable value is 64K.

The code required to tokenize these data items can get rather interesting. Since the macro processor only uses character data and the macro processor will not allow special characters in a macrovariable name, data transformation is often required. You may have to use the translate function to replace special characters in the lookup table's key variable data with characters acceptable as part of a macrovariable name. The underscore normally works well as a substitute value. Just use the same translation methodology when resolving the symbols in step two and the special character limitations should not be a problem. Tokenization of numeric data requires conversion to character data, as illustrated in the sample code. You can let SAS® convert it for you, but it's normally better to include a put function to translate it with a specific format to avoid unexpected results. After the concatenated macrovariable name and variable values to be tokenized have been translated as necessary, make sure there aren't any leading or embedded spaces in the strings. You can use the compress, catt, or other functions to prevent or eliminate these spaces.

Step Two: Recreate the "merged" variables in the destination dataset.

In a data step that will process the destination data set anyway, resolve the macrovariable symbols and recreate the variables that they came from in the lookup tables. Do not write a special data step for this step. The efficiency gains from this process are derived from the elimination of multiple read and write operations on the destination data set. Since the process of recreating the "merged" variables and populating them with source table data does not require a separate data step, don't waste the processing time by putting this task in a separate data step.

The example code is only "merging" numeric variables. When using this process to "merge" character data, it is necessary to include a length statement in this data step to insure that the character variable(s) created in the destination dataset are large enough to contain the tokenized data without truncation.

Since the NVar and NObs variable values were concatenated and tokenized into a single set of macrovariables, these values must be extracted from the concatenation using the scan function and then converted back to a numeric value with the input function.

```
/* This data step will recreate the merged
   variables in the destination data set. */
data work.V_Merge_Demo;
  set SASHELP.VColumn(keep = LibName -- Length);
  where Libname = "SASHELP";
  /* Create a single variable from a set of symbols. */
  Mod_Date = input(symget(catt("Mod_Date_",LibName,"_",MemName)),best.);
  format Mod_Date DateTime16.;
  /* Use the scan function to create multiple variables
     from a single set of symbols. */
  Nvar =
    input(scan(symget(catt("NObs_Data_",LibName,"_",MemName)),1,"~"),best.);
  NObs =
    input(scan(symget(catt("NObs_Data_",LibName,"_",MemName)),2,"~"),best.);
run;
```

Self Cleaning Capabilities

Obviously, this process can create a very large number of macrovariable symbols that take up memory. One of the benefits offered by the macro processor is the local symbol table that is automatically deleted when the macro completes its processing. By placing the entire virtual merge process into a macro, all of the macrovariables can be created in a local symbol table that will be automatically deleted when the macro completes execution.

CONCLUSION

The Virtual Merge greatly enhances processing efficiency by creating the same end result as merging any number of tables without the I/O intensive sorting, reading, writing, and resorting tasks that traditional merging requires. In fact, the data sets involved in the "virtual merge" process can be in a completely random order.

In order to be brief, this paper is limited to the basic concept of this process. The example in this paper only involves one master data set and one lookup table, but maximum benefits are achieved by using this method to replace many merges that would require repeated sorting by many different variables. As long as the symbols generated to contain the lookup table values conform to macrovariable naming rules, are distinct, and can be reconstructed from the data in the master data set, there are virtually no limits to the number of tables that can be "merged" in one data step. Replacing several traditional merges, and their associated sort procedures, with one "virtual merge" process can eliminate many hours of processing time.

ACKNOWLEDGMENTS

This process was developed several years ago after a "water cooler" discussion with Ron Richardson, one of my co-workers at Regions. Ron mentioned that it would save so much time if we could merge unsorted data. Although I wrote the code to make this work, Ron's idea is what got it all started.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Steve Sanders
Regions Financial
2050 Parkway Office Circle
Birmingham, AL 35244
Work Phone: (205) 560-7231
Fax: (205) 560 - 3593
E-mail: Steve.G.Sanders@Regions.com or Steve.G.Sanders@gmail.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.