CONCAT.SAS: A MACRO FOR CONCATENATION OF GROUPED MULTI-RECORD NARRATIVE TEXT
Roy B. Henrichs, S.F. Bay Area Rapid Transit District, Oakland, California

ABSTRACT
This paper presents a SAS macro that concatenates grouped multi-record fixed fieldwidth narrative text into a new field keyed to the group. An output SAS dataset contains two fields: 1) A group identifier. 2) The concatenated narrative output.

Multi record text concatenation may be required to enable the researcher to perform string searches and text-based data mining across multiple records, while preserving substring context within the total concatenated string. Concatenation process efficiency is critical, considering both DASD utilization and CPU time, when performing text concatenation on large datasets.

Specifically, this paper presents a SAS macro that the Reliability Engineering Department of the San Francisco Bay Area Rapid Transit District uses to concatenate multi-record text contained in the Maintenance and Reliability Information System (MARIS), a DB2 application that contains transit system operation and maintenance history. While this paper focuses on operation of the CONCAT.SAS macro, summary details are furnished for the total process. Also, we show how this approach has broad application in the fields of research and data mining.

INTRODUCTION
The San Francisco Bay Area Rapid Transit District (BART) operates a heavy rail rapid transit system that is located in the greater San Francisco Bay Area. Currently, rail lines extend from Colma in the west to Pittsburg/Bay Point in the east, with an extension currently under construction to the San Francisco International Airport and to Millbrae. Also, in the East Bay, lines run from Richmond in the north, to Fremont in the south; an additional line runs to Dublin/Pleasanton in the east.

Daily system operation requires an extensive infrastructure of transit vehicles (rail cars), elevators, escalators, fare collection systems, train control equipment, and related systems that must be maintained in good operating order. It is likely that system equipment requirements will increase in the future, with continued ridership growth and expansion of the core system.

A mainframe DB2 application called the Maintenance and Reliability Information System (MARIS) serves as a data warehouse for all equipment maintenance data. In MARIS, a GUI provides users the ability to enter data into the system, to perform certain standard data retrievals, and to run reports.

BART’s Reliability Engineering Department uses MARIS data to understand patterns in equipment maintenance history, and to make recommendations that can result in improved equipment reliability and availability. Currently, data retrieval and analysis is done using SAS/Connect and SAS/Access to DB2. Typically, SQL queries are prepared and submitted using the PC SAS client Query window. In most cases, multiple tables are joined, and a subset of data taken from the data warehouse for subsequent analysis.

THE PROBLEM
The format of narrative text contained in MARIS for each incident was found unsuitable for analysis, in raw form. Specifically, this resulted from the structure of the data itself.

Data Structure
MARIS narrative text for a single incident typically spans multiple records, with the total number of records being user-defined at the time of data entry. The maximum number of records per incident is unlimited, but always equals or exceeds one record. Narrative text is entered into a fixed-width field that uses trailing blanks as filler.

Each record contains the following data:
- Incident number (key field).
- Narrative number (subkey field) – varies from 1 to “n”, no upper bound.
- Narrative ($1000. fixed format, blanks appended as padding)

Implications for Users
While this multiple record format is convenient for data entry purposes, it poses a variety of problems to the end user. First, the record spanning narrative is difficult for users to read. Second, users have no easy way to concatenate text records together using “point-and-click” methods like the Query window or Enterprise Guide. Third, since string context can be lost between records, simple string searches and text-based data mining may not produce the intended results.

THE SOLUTION
One way to eliminate these problems is to generate a concatenated narrative for each value of the key field. Then, users can access the output concatenated field easily, using point-and-click tools. Finally, since concatenation produces a single string output, context is preserved within the output text. This protects the integrity of string search and text-based data mining efforts.

Since the underlying data is in DB2 format, it was felt that a native DB2 solution would be ideal. However, after consulting BART’s DB2 programming staff, it was found that DB2 did not provide a direct means for accomplishing this task. So, a good alternative appeared to be development of a SAS autocall macro that would produce concatenated text output linked to a key field.

Under SAS Version 8, the maximum allowable fieldwidth is 32,000 characters. A preliminary study revealed that, in most cases, total concatenated fieldwidth typically would not exceed 1,000 characters. Therefore, fieldwidth under Version 8 was not considered a constraint. On the other hand, this approach would not have been feasible under SAS Version 6, due to its 200 character fieldwidth limitation.
MACRO DESIGN
This work resulted in development of an autocall macro called CONCAT.SAS. Input to the macro consists of an input dataset, with user-specified key, subkey field, and narrative fields. Output from the macro consists of an output dataset, with user-defined key and concatenated narrative fields.

The CONCAT.SAS macro uses two additional autocall macro programs that perform independent utility functions. These macros are called OBSNVAR.SAS and VLENMAX.SAS.

OBSNVAR.SAS is used to determine the number of variables in a dataset - a key input to CONCAT.SAS. VLENMAX.SAS is used to determine the maximum fieldwidth (excluding any trailing blanks) occupied by text in a fixed width field that is padded with trailing blanks.

Basic Concepts
The SAS language permits concatenation of multiple fields into a single field using the concatenation operator within a data step.

That is:

concarr=narr1||narr2||...||narrn;

where narr1 through narrn contain narrative text to be concatenated into a new variable called concarr, and "n" is the total number of variables.

Note that variables narr1 through narrn must be in a logical order for this process to work. That is, if two narratives to be concatenated are "Mary had a little lamb", and "its fleece was as white as snow", then the order of concatenation is important, and must be maintained.

In the current case, we have a key variable, subkey variable, and a narrative text variable. Since the desired output is a key variable and a concatenated text variable, we begin the process by sorting the narrative text data by key and subkey. This ensures that the narrative text data is in logical order for each value of key. Then, the subkey variable is no longer needed, and can be dropped.

Next, we transpose the data, to produce a dataset containing variables key and narr1 through narrn. Here, we create variables narr1 through narrn that contain the value of input variable narr for each value of the key variable:

proc transpose data=indata
   out=transdata
   prefix=narr;
by key;
var narr;
run;

Here, the "var" statement defines the input variable to be transposed (narr); the "prefix=narr" statement defines the prefix of the output variable as "narr" (i.e., narr1, narr2, ..., narrn). Thus, the OUT dataset contains a series of narrative variables (in appropriate order) that we can concatenate, plus the key variable.

Trailing Blanks
Performing simple concatenation on the narrative text in the current case would result in unacceptable output, because trailing blanks are used to fill a total fieldwidth of 1000 characters. Following the simple example, if narr1="Mary had a little lamb <978 blanks>", and narr2=" its fleece was as white as snow <969 blanks>", then the concatenated narrative output would contain 978 blanks in the middle, and the total concatenated string length would be 2,000 characters!

Fortunately, SAS provides a simple solution to this problem: the TRIM function, which eliminates trailing blanks. So, instead of concatenating the input narratives, we could use the TRIM function to eliminate trailing blanks:

concarr=trim(narr1)||trim(narr2) ...
||trim(narrn);

But, this introduces another problem. Since trailing blanks are eliminated, this causes narrative text to run together. Continuing with the above example, the resultant concatenated output (concarr) would be "Mary had a little lamb its fleece was as white as snow". Note that "lamb" and "its" are run together, because all trailing blanks are deleted - thus producing the unintended output "lambits".

This is unacceptable for most situations involving narrative (language) text. Blanks could be embedded after each narrative, but this, too, could produce incorrect results. Cases exist where a data input routines intentionally breaks words in the middle: for example, if input text exceeds (say) 1,000 characters, a word may be broken within the 1,000th and 1,001st characters. Therefore, it appeared that an option to use one or more trailing blanks at the time of concatenation is needed. We implement this within CONCAT.SAS in what one might call a "narrative delimiter string."

Narrative Delimiter Strings
As stated above, a narrative delimiter string should provide the ability to incorporate or exclude one or more trailing blanks. Also, in some cases, it might be desirable to provide a fixed delimiter (defined at run time) between subsequent records (subkey) of the original data. So, it was decided that a "narrative delimiter" should be defined at run time.

As implemented in CONCAT.SAS, the narrative delimiter may consist of a single blank; a series of characters; or no characters at all ("null delimiter string" - a case that requires special handling). Thus, in the example, using the narrative delimiter '***', the concatenated narrative output becomes "Mary had a little lamb***its fleece was as white as snow". Alternately, use of a single space as a delimiter would result in syntactically correct output.

To enable capturing of blanks within the narrative delimiter, the narrative delimiter macro variable &NARRDLMS always includes leading and trailing single quotation marks. Continuing the example, to use the delimiter '***', we set &NARRDLMS to equal "***" (includes the single quotes).

Generation of Concatenated Output
A SAS macro language loop (%do) is used to generate a macro variable that concatenates narrative variables narr1 to narrn together in a subsequent data step. Since narrative delimiters are included in the macro variable, these will be included in the final output concatenated narrative variable. (Note that
If the delimiter string is &NARRDLM, then the desired output macro variable (&trimstr) is:

\[
&trimstr=trim(narr1) || &NARRDLM \\
|| trim(narr2) || &NARRDLM \\
| . . . | || trim(narrn) ,
\]

and can be used in a SAS data step that produces the concatenated narrative, as follows:

```sas
data work.concat; 
set work.transdata; 
narrout=&trimstr; 
run;
```

where work.transdata contains the transposed dataset output (that is, key and narr1 through narrn variables), and work.concat contains the output dataset (i.e., contains variables key and narrout - the concatenated narrative output).

With concatenation now complete, variables narr1 through narrn become redundant, and can be dropped from the output using a SAS data step. Also, additional post-processing is required to handle cases where (say) only two of n narrative variables contain non-blank text; such cases will have one or more trailing delimiter strings at the end of the concatenated narrative. Such trailing delimiter strings are removed using the TRANWRD function, in a SAS data step.

Number of Narratives: OBSNVARS.SAS Macro

Since the total number of narratives is needed to determine boundary conditions for the macro language loop, this must be determined at run time. The current work uses a SAS autocall macro called OBSNVARS.SAS that is based on code provided by SAS (1997), pp. 242-243. Other than causing the program to display the number of observations and variables to the SAS log, source code is virtually identical to that of SAS (1997).

As implemented herein, OBSNVARS.SAS produces three output global macro variables, one of which is the number of variables in a specific dataset (&nvars). Here, the number of variables is the same as the maximum number of narratives. Therefore, this is the result passed to CONCAT.SAS.

This macro uses %SYSFUNC as its basis; readers needing further details on this topic should consult SAS (1997).

Maximum Length of a Variable: VLENMAX.SAS Macro

It was recognized that maximum fieldwidth determination and adjustment would be needed at several points in CONCAT.SAS. So, an additional autocall macro called VLENMAX.SAS was developed.

Operation of this macro is fairly straightforward. For a specified dataset and variable, the macro determines maximum fieldwidth, excluding trailing blanks. First, we use the LENGTH function in a SAS data step to define a variable vlen:

\[
vlen=length(&var_in);
\]

where vlen is the length of the variable defined by macro variable &var_in, excluding trailing blanks. Then, PROC MEANS is used to determine the maximum value of variable vlen in the input dataset work.len:

```sas
proc means noprint data=work.len max; 
var vlen; 
output out=work.vlenmax 
MAX(vlen)=maxlen; run;
```

Here, the maximum value of vlen is saved in output dataset work.vlenmax.

Then, using a _null_ datastep and call symput, the value of maxlen is put into local macro variable &maxvlen:

```sas
data _null_; 
set work.vlenmax; 
call 
symput(‘maxvlen’,left(put(maxlen,9.))); 
run;
```

Finally, the value of local macro variable &maxvlen is put into global macro variable &vlenmax, the output of VLENMAX.SAS:

```sas
%let vlenmax; 
%global vlenmax; 
%let vlenmax=%eval(&maxvlen);
```

Within CONCAT.SAS, a data step uses &vlenmax to optimize fieldwidth, as follows:

```sas
data &fileout; 
retain &key &narrout; 
length &narrout $&vlenmax; 
set &filein; 
run;
```

Here, &filein is the input file, &fileout is the output file, &key is the key variable, and &narrout is the concatenated output narrative variable. Fieldwidth adjustment is done using the LENGTH statement, taking &vlenmax as input from VLENMAX.SAS.

DASD Utilization and Computing Time

Since the current work uses fixed-width fields in a large mainframe DB2 database, programming efforts should attempt to optimize DASD utilization and computing time.

Attempts have been made to minimize fieldwidth for temporary and final output datasets. Specifically, VLENMAX.SAS is used at the first step of the CONCAT.SAS macro, to minimize DASD use for all subsequent temporary SAS datasets. Finally, it is used at the last step to minimize fieldwidth for the final output dataset.

While efforts were made to minimize the use of temporary datasets, such datasets are necessary at various stages of the concatenation process. To minimize the use of temporary workspace, PROC DATASETS is used throughout the CONCAT.SAS macro, to delete these datasets when they are no longer needed.
Use of these methods resulted in better than a threefold improvement in performance of the total program. But, as program development progressed, it soon became evident that marginal improvements in resource utilization trade off with resources needed to implement such improvements. In some instances, this may be a function of dataset size. And, finally, there is the issue of program complexity required to accomplish this.

Given these issues, it appears likely that there is considerable ambiguity in defining an "optimal" program configuration.

MACRO IMPLEMENTATION AND USE
Currently, BART Reliability Engineering uses the CONCAT.SAS macro for concatenation of MARIS incident narratives. Specifically, the CONCAT.SAS macros is run against an existing MARIS DB2 table, to produce a second DB2 table that contains a key field and concatenated incident narrative.

Work has begun to enable automatic updating of the concatenated narrative table during periods of low system utilization (i.e., 4 A.M.). Periodic updating of the table will be performed during the day using transaction records, so that rapid updates can be obtained. Then, resultant tables can be accessed via SAS/Connect and SAS/Access to DB2, using the Query window. Also, plans are being made to download a version of the table to a Windows NT server, for subsequent processing and routine analysis.

While BART currently uses this macro to support analysis of system-wide maintenance data, it should prove useful in a variety of other applications. Since the macro is generic in nature, its use is not restricted to either transit-specific applications, or mainframe databases.

Recently, the author has encountered needs for this program in the field of marketing research – specifically, an application requiring concatenation of open-ended survey data gathered via the Internet. It is likely that similar applications exist in a large number of fields of research.

To date, the program has been tested under SAS Version 8 under IBM OS-390, and also on a Windows 98 PC platform. While the program should run satisfactorily on other platforms, no testing has been performed on systems other than those specified.

FURTHER RESEARCH
Further refinement of this program is possible. Additional work to improve efficiency of the total process could be undertaken, possibly resulting in enhanced performance.

Algorithms could be developed to cause the program to adapt its optimization methods automatically to the size of the dataset to be processed. Also, other methods could be used in certain areas of the program, possibly involving a complex interface between SAS Macro language and SQL. Narrative delimiter string implementation could be expanded to provide for numbering of delimiters in the concatenated output, for situations where this is considered useful.

Finally, further optimization of the PROC TRANSPOSE routine could yield a performance improvement for datasets that do not contain at least one variable label. The original focus of the CONCAT.SAS macro design was concatenation of DB2 tables that contain variable labels. PROC TRANSPOSE generates a variable called _label_ whenever variable labels exist. To maintain consistency for datasets that do not contain variable labels, CONCAT.SAS creates an artificial variable label that is deleted after transposition. For users who routinely work with data that does not contain variable labels, elimination of this unnecessary variable label could save additional resources. Thus, further work to optimize the routine may be warranted in such instances.

SOURCE CODE AVAILABILITY
While this paper discusses the basic ideas involved in design and operation of the CONCAT.SAS macro, source code is not provided due to its length. Part of this has resulted from extensive comments interspersed throughout the code. However, source code can be obtained from the author on request.

CONCLUSION
This paper has presented a SAS macro-based approach to concatenation of grouped multi-record fixed fieldwidth narrative text. While the macro was prepared for use in analysis of an rapid transit system equipment maintenance database on a mainframe computer, it has proven useful for general marketing research and text-based data mining applications.

ACKNOWLEDGMENTS
I would like to thank Ms. Catherine Lee and Mr. Victor Liu, of the BART Information Technology Department, for their help in implementing this program within the context of MARIS on the District's IBM mainframe computer, and for their ongoing support in this project. Also, my thanks to Mr. Chuck Church of BART Information Technology Department for suggesting that this material be prepared for presentation at WUSS.

REFERENCES

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

Dr. Roy B. Henrichs, P.E.
Manager of Reliability Engineering
S.F. Bay Area Rapid Transit District
Reliability Engineering Department
1000 Broadway, Suite 600
Oakland, CA 94607
Work Phone: (510) 464-6870
Fax: (510) 464-6838
E-mail Address: rhenric@bart.gov , rovh@pacbell.net