Purrfect Functions For Data Concatenation

Presented To The Edmonton SAS User Group
April 24, 2018
By John Fleming

We Have A Problem

In a database, we have four variables for a person's name as follows:

Varname	Length
Last	$40
First	$20
Middle1	$20
Middle2	$20

The data in these variables is kind of/sort of messy because the data is often imported with leading blanks.

We want to create a new variable, Full, that contains the person's name in the form –

```
Full = "Last, First Middle1 Middle2"
```

For Example

Varname Value

```
Last      “     Schmitt                    “
First     “John                “
Middle1   “   Jacob             “
Middle2   “  Jingleheimer       “
```

Combine the variables so we get this –

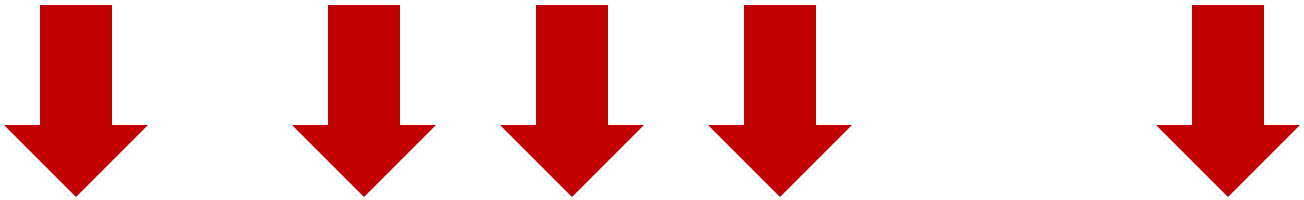
```
Full = “Schmitt, John Jacob Jingleheimer”
```

The Obvious First Approach

```
Full = Last||',',||First||Middle1||Middle2;
```

The Problem With The Obvious First Approach

There are a **lot** of extra spaces



Full = “ Schmitt , John Jacob Jingleheimer “

We Could Add Lots And Lots Of Functions

Use “left” and “trim” to remove leading and trailing blanks

Insert a comma and a blank space between the last name and the first name.

```
Full = trim(left(Last))||', ' ||trim(left(First))||
       ' '||trim(left(Middle1))||' '||trim(left(Middle2));
```

Insert just a space between the first name and the middle names after removing leading and trailing blanks..

Is There An Easier Way?

Well yes, there is and easier way.

```
Full = catx(' ', cats(Last, ','), First,
            Middle1, Middle2);
```

These two functions might be new to some -

```
Full = catx(' ', cats(Last, ','), First,
            Middle1, Middle2);
```

CATX and CATS are two members of a family of feline concatenation (or CAT) functions and call routines

- CAT
- CATS
- CATT
- CATX
- CATQ
- Call CATS
- Call CATT
- Call CATX

Cat Syntax – The CAT, CATS, and CATT Functions

These three functions have the same syntax -

```
NewVar1 = CAT(item-1 <, ..., item-n>);
NewVar2 = CATS(item-1 <, ..., item-n>);
NewVar3 = CATT(item-1 <, ..., item-n>);
```

Cat Syntax – The CAT, CATS, and CATT Functions

```
NewVar1 = CAT(item-1 <, ..., item-n>);
NewVar2 = CATS(item-1 <, ..., item-n>);
NewVar3 = CATT(item-1 <, ..., item-n>);
```

The difference between these three functions is how they handle leading and trailing blanks in the source items.

For example, CAT returns a concatenated character string without removing any leading or trailing blanks.

```
NewVar = CAT(item-1 <, ..., item-n>);
```

Is equivalent to:

```
NewVar = item-1||Item-2|| ... ||item-n;
```

If we name our variables right, we can make the CAT syntax even simpler and more concise

```
NewVar1 = cat(of x1–x4);
```

The CATS function removes all leading and trailing blanks, and returns a concatenated character string.

```
NewVar = CATS(OF X1-X4);
```

Is equivalent to:

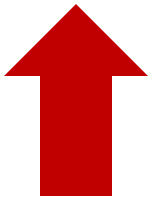
```
NewVar = TRIM(LEFT(X1))||
         TRIM(LEFT(X2))||
         TRIM(LEFT(X3))||
         TRIM(LEFT(X4));
```

Comparing CAT with CATS

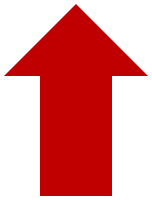
```
var = ' I am an evil test variable      ';

     new_var1 = cat('[',var,']');
     new_var2 = cats('[',var,']');

New_var1 = "[ I am an evil test variable      ]"
New_var2 = "[I am an evil test variable]"
```



CAT leaves the blanks in while CATS takes the blanks out



The CATT function removes trailing blanks only, and returns a concatenated character string

```
NewVar = CATT(OF X1-X4);
```

Is equivalent to:

```
NewVar = TRIM(X1)||TRIM(X2)||
         TRIM(X3)||TRIM(X4);
```

Cat Syntax – The CATX Function

```
NewVar = CATX(delimiter, item-1 <,...
         item-n>)
```

The CATX removes all leading and trailing blanks, inserts delimiters, and returns a concatenated character string.

```
NewVar = CATX(SP, OF X1-X4);
```

Is equivalent to:

```
NewVar = TRIM(LEFT(X1))||SP||
         TRIM(LEFT(X2))||SP||
         TRIM(LEFT(X3))||SP||
         TRIM(LEFT(X4));
```

Where SP is a quoted character (for example, ‘,’), a quoted character string (for example, ‘likes to eat’ or a variable containing the delimiter.

In our original challenge, CATX inserts the blanks between the variables.

```
Full =   catx(' ', cats(Last, ','),
         First, Middle1, Middle2);
```

produces

```
Full =   "Schmitt, John Jacob
         Jingleheimer"
```

Cat Syntax – The CATQ Function

```
NewVar = CATQ(modifiers<, delimiter>,
          item-1 <, ..., item-n>);
```

CATQ concatenates character or numeric values by using a delimiter to separate items and by adding quotation marks to strings that contain the delimiter. For example:

For example, source data

Lawyer	Firm
Andrew Jones	Jones, Jones, Jones & Jones LLP
Belinda Jones	Jones, Jones, Jones & Jones LLP
Charles Jones	Jones, Jones, Jones & Jones LLP
Daphne Jones	Jones, Jones, Jones & Jones LLP

NewVar = CATQ('2D',', ', Lawyer, Firm); produces-

NewVar

Andrew Jones, "Jones, Jones, Jones & Jones LLP"
Belinda Jones, "Jones, Jones, Jones & Jones LLP"
Charles Jones, "Jones, Jones, Jones & Jones LLP"
Daphne Jones, "Jones, Jones, Jones & Jones LLP"

The CATQ modifier allows us to specify –

- Whether to use single or double quote marks
- Delimiter options
- Whether or not to strip leading and/or trailing blanks before concatenation

Length Of Output Matters

CAT functions return values of the following lengths:

- up to 200 characters in WHERE clauses and in PROC SQL
- up to 32,767 characters in the DATA step except in WHERE clauses (approximately 2^8)
- up to 65,534 characters when CATX is called from the macro processor (approximately 2^9)

Check Out The Cat Calls

The three call routines

- CALL CATS
- CALL CATT
- CALL CATX

Further Reading

"Purrfectly Fabulous Feline Functions"
Louise S. Hadden, Abt Associates Inc.
SAS Global Forum 2010

Paper 205-2010

SAS Global Forum 2010 Posters

Paper 205-2010

Purrfectly Fabulous Feline Functions
Louise S. Hadden, Abt Associates Inc.

ABSTRACT

Explore the fabulous feline functions and calls available in SAS® 9.1 and later. Using CAT functions and CAT CALLs gives you an easier way to streamline your SAS code and facilitate concatenation of character strings. So, leave verbose coding, myriad functions, and the vertical bar concatenation operators behind! SAS® 9.2 enhancements will also be demonstrated.

INTRODUCTION

```
LENGTH stcounty $ 5 citystzip $ 60;

stcounty=TRIM(LEFT(PUT(state,Z2.)))||TRIM(LEFT(PUT(county,Z3.)));

citystzip=TRIM(LEFT(city))||', '||TRIM(LEFT(statecode))||' '||TRIM(LEFT(zip));
```

Version 9 introduced four new functions (CAT, CATS, CATT, and CATX) and three new CALL routines (CALL CATS, CALL CATT, and CALL CATX) that replace the clunky syntax shown above, and add some additional enhanced capability to character string concatenation. In Version 9.2, CATQ was added. We will explore the five (fabulous) CAT functions and three CAT CALLs and demonstrate how string concatenation can be accomplished more easily and efficiently using these functions and call routines. In addition, the differences between the CAT functions and CAT CALLs will be briefly discussed. Examples were run with SAS 9.2 on Windows XP, and SAS 9.2 on Windows Server (x64). Leaving appropriate spaces out is deliberate so that the action of the functions and CALL routines can be observed. Performance using the concatenation operator, CAT functions and CAT CALL routines will be compared on the two different platforms using an identical one million record file.

ORIGINS OF CAT CALLS AND FUNCTIONS

The customary syntax seen above, varx=TRIM(LEFT(vary))) || TRIM(LEFT(varz))), consists of two functions (TRIM and LEFT) paired with the concatenation operator (|| .) TRIM removes trailing blanks, while LEFT left aligns text values. LEFT does not remove leading blanks. It simply moves leading blanks to the end of the string, which is why you usually see LEFT used in conjunction with (and inside) the TRIM function for string concatenation. Over the years, SAS has enhanced the function (pardon the pun) of TRIM and LEFT. A newer function TRIMN also strips trailing blanks: the difference between TRIM and TRIMN is the end result in the case of the argument being missing. TRIM results in a single blank (length 1) for a missing argument while TRIMN results in a null (length 0.) This might come in handy, for example, when concatenating first, middle and last names, in which middle names are frequently missing. Another newer function, STRIP, is the equivalent of TRIMN(LEFT(varx)) but is obviously more convenient. STRIP removes both leading and trailing blanks. The CAT CALLS and functions build on the original and derivative functions to offer a complete array of concatenation options.

All examples shown below were run on a Windows X64 server using SAS version 9.2, on one million records. In this case, all arguments being concatenated are character, and the second (of three) arguments is blank. Note that it is not necessary for all arguments to be character with the CAT CALLs and functions, and that arguments may be of mixed type. In addition, results of CAT functions (but not CAT CALLs) may be numeric. Unlike with the concatenation operator, you will not get a warning in your log if you use numeric arguments with the CAT CALLs and functions.

1

Questions?

John Fleming
Alberta Health Services

(780) 643 – 4341
John.fleming@albertahealthservices.ca