PROC SQL

How can I stop certain functions setting the length of the result to 200?

In the Language of SAS there are essentially three lengths which character manipulation functions will return:

- 8 bytes
- 200 bytes
- Inherited from the source

Where the system can make a definitive decision of the length of the result at Compile Time it will do so, however there are some oddities - some functions provide additional information, others do not.

Consider the code:

```
data test ;
  nums = repeat('1234567890', 25) ;
run ;
```

In this example the system is unable to identify the length, and therefore sets the length of the variable to 200, causing a truncation of the result. No notification is provided. Where a result will be greater than 200, ensure an explicit length is set.

```
data test ;
  length nums $ 256 ;
  nums = repeat('1234567890', 25) ;
run ;
```

Using this variable as a basis consider the following character manipulations:

```
proc sql ;
  create table trunc as
  select  trim(nums)                  as trim
         ,strip(nums)                 as strip
  ,right(nums)                  as right
  ,substr(nums,2,20)            as substr
         ,translate(nums,'~#','50')   as translate
  ,tranwrd(nums,'0','#-#')      as tranwrd
         ,scan(nums,1,'0')            as scan
         ,cats(nums,'###')            as cats
         ,prxchange('s/0/@/',-1,nums) as prxchange
         ,case when ranuni(0) > 0
```

```
           then cats(nums,'###')
         end                          as case
  from test
  ;
quit;
```

TRIM, STRIP, RIGHT, SUBSTR, and TRANSLATE all inherit their length from the source. (Although the length of the SUBSTR variable might be derived from the third argument, this is optional and is not always present.)

TRANWRD, SCAN, CATS, and PRXCHANGE if not previously set will take a default length of 200. They act differently however: TRANWRD, SCAN and PRXCHANGE still perform the character manipulation, then (potentially) truncate the result. CATS generates a WARNING that the result, "may either be truncated to 200 character(s) or be completely blank".

```
WARNING: In a call to the CATS function, the buffer allocated for the
result was not long enough
         to contain the concatenation of all the arguments. The correc
t result would contain 259
         characters, but the actual result may either be truncated to
200 character(s) or be
         completely blank, depending on the calling environment. The f
ollowing note indicates the
         left-most argument that caused truncation.
```

To resolve the issue an explicit length can be set:

```
proc sql ;
  create table no_trunc as
  select  trim(nums)                  as trim      length = 280
         ,strip(nums)                 as strip     length = 280
  ,right(nums)                 as right     length = 280
         ,substr(nums,2,20)           as substr    length = 280
         ,translate(nums,'~#','50')   as translate length = 280
  ,tranwrd(nums,'0','#-#')     as tranwrd   length = 280
         ,scan(nums,1,'0')            as scan      length = 280
         ,cats(nums,'###')            as cats      length = 280
         ,prxchange('s/0/@/',-1,nums) as prxchange length = 280
         ,case when ranuni(0) > 0
            then cats(nums,'###')
          end                         as case      length = 280
  from test
  ;
```

```
quit;
```

While this works for most of these scenarios, the WARNING is still generated for the CATS function within the CASE expression.

Looking at the metadata for the dataset the length appears to have been set, but the character manipulation is not performed.

```
Alphabetic List of Variables and Attributes

#    Variable     Type    Len

10    case         Char    280
 8    cats         Char    280
 9    prxchange    Char    280
 3    right        Char    280
 7    scan         Char    280
 2    strip        Char    280
 4    substr       Char    280
 5    translate    Char    280
 6    tranwrd      Char    280
 1    trim         Char    280
```

Care should be taken that the lengths of character variables are correctly set at Compile Time to avoid either incorrect truncation, or indeed excess 'padding' of character variables.

Unique solution ID: #1015
Author: Alan D Rudland
Last update: 2017-03-27 16:46