Paper BtB-16

Same Data Different Attributes: Cloning Issues with Data Sets

Brian Varney, Experis Business Analytics, Portage, MI

ABSTRACT

When dealing with data from multiple or unstructured data sources, there can be data set variable attribute conflicts. These conflicts can be cumbersome to deal with when developing code. This paper intends to discuss issues and table driven strategies for dealing with data sets with inconsistent variable attributes.

INTRODUCTION

This paper intends to provide information to SAS® programmers of all levels and help them resolve issues when combining data that has content that is appropriate for combining but variable attributes that are not identical. Depending on the situation with your data and the method you are using to combine the data, inconsistent variable attributes can significantly affect the results from your program. This paper will also show you how to set up a data set with the desired attributes and then use it as the standard template for others programmatically.

One disclaimer for this paper is that at the time this paper was written, I only had brief exposure to SAS 9.4. There could be features and functionality that I am not familiar with.

DATA SET AND VARIABLE ATTRIBUTES

SAS data set and variable attributes can be viewed by examining results from a PROC CONTENTS. Consider the program and results shown below. The PROC CONTENTS results are show in 4 figures based on the how the output is organized. For the purpose of this paper, I have used SASHELP.ZIPCODE. It should be readily available within any SAS environment.

```
proc contents data=sashelp.zipcode;
run;
```

Data Set Name	SASHELP.ZIPCODE	Observations	41267
Member Type	DATA	Variables	19
Engine	V9	Indexes	1
Created	06/20/2013 01:37:16	Observation Length	816
Last Modified	06/20/2013 01:37:16	Deleted Observations	0
Protection		Compressed	NO
Data Set Type		Sorted	YES
Label	US Zipcodes; Source: zipcodedownload.com Jan 2013, SAS Release 9.3		
Data Representation	WINDOWS_64		
Encoding	us-ascii ASCII (ANSI)		

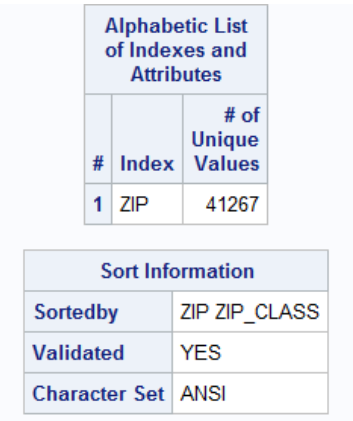
Figure 1. Top Section of SAS Data Set Attributes from PROC CONTENTS Output

Engine/Host Dependent Information	
Data Set Page Size	65536
Number of Data Set Pages	517
First Data Page	1
Max Obs per Page	80
Obs in First Data Page	75
Index File Page Size	4096
Number of Index File Pages	186
Number of Data Set Repairs	0
ExtendObsCounter	YES
Filename	C:\Program Files\SASHome94\SASFoundation\9.4\core\sashelp\zipcode.sas7bdat
Release Created	9.0401B0
Host Created	X64_7PRO

Figure 2. Second Section of SAS Data Set Attributes from PROC CONTENTS Output

Alphabetic List of Variables and Attributes					
#	Variable	Type	Len	Format	Label
18	ALIAS_CITY	Char	300		USPS - alternate names of city separated by ||
19	ALIAS_CITYN	Char	300		Local - alternate names of city separated by ||
12	AREACODE	Num	8		Single Area Code for ZIP Code.
13	AREACODES	Char	12		Multiple Area Codes for ZIP Code.
5	CITY	Char	35		Name of city/org
9	COUNTY	Num	8		FIPS county code.
10	COUNTYNM	Char	25		Name of county/parish.
16	DST	Char	1		ZIP Code obeys Daylight Savings: Y-Yes N-No
15	GMTOFFSET	Num	8		Diff (hrs) between GMT and time zone for ZIP Code
11	MSA	Num	8		Metro Statistical Area code by common pop-pre 2003; no MSA for rural
17	PONAME	Char	35		USPS Post Office Name: same as City
6	STATE	Num	8		Two-digit number (FIPS code) for state/territory
7	STATECODE	Char	2		Two-letter abbrev. for state name.
8	STATENAME	Char	25		Full name of state/territory
14	TIMEZONE	Char	9		Time Zone for ZIP Code.
3	X	Num	8	11.6	Longitude (degrees) of the center (centroid) of ZIP Code.
2	Y	Num	8	11.6	Latitude (degrees) of the center (centroid) of ZIP Code.
1	ZIP	Num	8	Z5.	The 5-digit ZIP Code
4	ZIP_CLASS	Char	1		ZIP Code Classification:P=PO Box U=Unique zip used for large orgs/businesses/bldgs Blank=Standard/non-unique

Figure 3. Section of Variable Attributes from PROC CONTENTS Output



Alphabetic List of Indexes and Attributes

#	Index	# of Unique Values
1	ZIP	41267

Sort Information	
Sortedby	ZIP ZIP_CLASS
Validated	YES
Character Set	ANSI

Figure 4. Bottom Section with Additional Data Set Attributes from PROC CONTENTS Output

SAS DATA SET ATTRIBUTES

Referencing the output shown in the four previous figures, the SAS data set attributes are shown in figures 1, 2, and 4. They are not the focus of this paper but they have other issues depending on how you are working with SAS Data Sets.

One important issue with SAS Data Set attributes is that they do not persist if you re-create a SAS Data Set during processing. Consider the code below…

```
data zipcodes;
   set zipcodes;
run;

Or

proc sql;
   create table zipcodes as
   select *
   from zipcodes;
quit;
```

If you were to run the above code, any data set attributes that you have added such as data set labels, indexes, constraints, flags that indicate the data set has been sorted, etc. will be lost.

SAS VARIABLE ATTRIBUTES

Figure 3 from shows the variable attributes from the SAS Data Set SASHELP.ZIPCODES that are displayed by PROC CONTENTS. This paper will be focusing on the variable attributes type, length, and format but to be complete I have leveraged SASHELP.VCOLUMN to show all of the variable attributes contained in the SAS Dictionary Tables.

```
proc print data=sashelp.vcolumn;
where libname='SASHELP' and memname='ZIPCODE';
run;
```

This information can also be leveraged from DICTIONARY.COLUMNS if you are using PROC SQL.

```
proc sql;
   select *
   from dictionary.columns
   where libname='SASHELP' and memname='ZIPCODE';
quit;
```

The data set SASHELP.VCOLUMN is actually a view that points at DICTIONARY.COLUMNS. Additionally, SASHELP.VTABLE and DICTIONARY.TABLES contain information about the SAS data set attributes. There are many other dictionary tables. You can see the list of all dictionary tables by querying DICTIONARY.DICTIONARIES.

To learn more about the SAS Dictionary Tables, please reference the SAS web site.

http://support.sas.com/documentation/cdl/en/lrcon/64801/HTML/default/viewer.htm#p11h9yhja6t25an1mkx8xgy3wcwm.htm

Obs	libname	memname	memtype	name	type	length	npos	varnum	label	format	informat	idxusage	sortedby	xtype	notnull	precision	scale	transcode
2785	SASHELP	ZIPCODE	DATA	ZIP	num	8	0	1	The 5-digit ZIP Code	Z5.		SIMPLE	1	num	no	.	.	yes
2786	SASHELP	ZIPCODE	DATA	Y	num	8	8	2	Latitude (degrees) of the center (centroid) of ZIP Code.	11.6			0	num	no	.	.	yes
2787	SASHELP	ZIPCODE	DATA	X	num	8	16	3	Longitude (degrees) of the center (centroid) of ZIP Code.	11.6			0	num	no	.	.	yes
2788	SASHELP	ZIPCODE	DATA	ZIP_CLASS	char	1	64	4	ZIP Code Classification:P=PO Box U=Unique zip used for large orgs/businesses/bldgs Blank=Standard/non-unique				2	char	no	.	.	yes
2789	SASHELP	ZIPCODE	DATA	CITY	char	35	65	5	Name of city/org				0	char	no	.	.	yes
2790	SASHELP	ZIPCODE	DATA	STATE	num	8	24	6	Two-digit number (FIPS code) for state/territory				0	num	no	.	.	yes
2791	SASHELP	ZIPCODE	DATA	STATECODE	char	2	100	7	Two-letter abbrev. for state name.				0	char	no	.	.	yes
2792	SASHELP	ZIPCODE	DATA	STATENAME	char	25	102	8	Full name of state/territory				0	char	no	.	.	yes
2793	SASHELP	ZIPCODE	DATA	COUNTY	num	8	32	9	FIPS county code.				0	num	no	.	.	yes
2794	SASHELP	ZIPCODE	DATA	COUNTYNM	char	25	127	10	Name of county/parish.				0	char	no	.	.	yes
2795	SASHELP	ZIPCODE	DATA	MSA	num	8	40	11	Metro Statistical Area code by common pop-pre 2003; no MSA for rural				0	num	no	.	.	yes
2796	SASHELP	ZIPCODE	DATA	AREACODE	num	8	48	12	Single Area Code for ZIP Code.				0	num	no	.	.	yes
2797	SASHELP	ZIPCODE	DATA	AREACODES	char	12	152	13	Multiple Area Codes for ZIP Code.				0	char	no	.	.	yes
2798	SASHELP	ZIPCODE	DATA	TIMEZONE	char	9	164	14	Time Zone for ZIP Code.				0	char	no	.	.	yes
2799	SASHELP	ZIPCODE	DATA	GMTOFFSET	num	8	56	15	Diff (hrs) between GMT and time zone for ZIP Code				0	num	no	.	.	yes

Figure 5. Listing from SASHELP.VCOLUMN for the SAS Data Set SASHELP.ZIPCODE

Now that we have looked at all of the SAS data set and variable attributes, let us re-focus to the variable attributes of interest for this paper (type, length, and format). When combining SAS Data Sets with inconsistent type, length, and format definitions, results can be compromised. Next we will discuss the methods for combining SAS Data Sets that are relevant to this paper.

METHODS FOR COMBINING SAS DATA SETS

At this point it is appropriate to discuss some of the different methods for combining SAS data sets. To start off with, let's discuss the three different scenarios for combining data; adding rows, adding columns, and modifying existing values. Please note that there are probably more methods available than discussed in this paper. We will mention three different data combination scenarios but in this paper we will focus on the Adding Rows scenario.

Modifying Existing Data Values

This type of operation is typically a master transaction type of update. You have two data sets. The first data set has all of the data and the other has some records with values that are meant to update a few values in the first data set by using primary keys to indicate which records are updated.

Adding Columns

This type of operation is typically a join/merge operation. You have two or more data sets with different variables except for the keys that are appropriate for matching up the records. In SAS terms this is known as a Data Step Merge or a PROC SQL Join.

Adding Rows

This type of operation is typically a stacking operation. You have multiple data sets with the same variables and you want to stack them. In SAS terms this is known as a Data Step SET, PROC APPEND, or PROC SQL OUTER UNION CORRESPONDING.

COMBINING SAS DATA SETS WITH NON-SAS DATA

Although the focus of this paper is not how to read in data from non-SAS data sources, it is worth mentioning the potential issues with some of the non-SAS sources.

EXCEL

Although an Excel worksheet looks and feels like a table, the attributes for a column do not necessarily need to be consistent. Attributes can change on a cell to cell basis thus making data extracts somewhat unpredictable. By default, SAS uses the first eight rows of data in a column to determine the optimal variable attributes.

ASCII / EBCDIC

Although ASCII / EBCDIC data does not have built in data set or variable attributes, you have complete control over how the data is read in. By using INFILE and INPUT statements, you can control the name, type, length and format of each variable.

OTHER STRUCTURED DBMS

This is the best non-SAS data to read in as structured databases enforce column attributes like in SAS data sets.

CREATING A TEMPLATE DATA SET

A useful approach in combining data sets with like variables but potentially inconsistent attributes is to create a SAS data set that has the desired data set and variable attributes but without any actual data records. There are a few methods for achieving this in SAS.

MANUAL METHODS

Using the SAS Data Step, you can create an empty SAS Data Set with the exact desired variable attributes.

Data Step Method

```
data class;
   attrib name   length=$8 label='Name'
          sex    length=$1 label='Sex'
          age    length=8  label='Age'        format=3.
          height length=8  label='Height CM'  format=5.1
          weight length=8  label='Weight lbs' format=5.1
   ;
   stop;
run;
```

PROC SQL Method

```
proc sql;
   create table class
   (name   char(8) label='Name',
    sex    char(1) label='Sex',
    age    num     label='Age'        format=3.,
    height num     label='Height CM'  format=5.1,
    weight num     label='Weight lbs' format=5.1
    );
quit;
```

AUTOMATICALLY BY USING ANOTHER SAS DATA SET

If there is another SAS data set which has the desired attributes, it is not efficient to re-invent the wheel as we did in the previous section. It is more efficient to point at another SAS Data Set that has the variable attributes that we desire.

Data Step Method

```
data zipcode;
   if 0 then
      set sashelp.zipcode;
   stop;
run;
```

PROC SQL Method

```
proc sql;
   create table class like sashelp.zipcode;
quit;
```

HANDLING ATTRIBUTE INCONSISTENCIES WHILE APPENDING

Since the focus of this paper is stacking SAS data sets with the same variables, this section will only focus on Data Step Set, PROC APPEND, and PROC SQL OUTER UNION CORRESPONDING.

VARIABLE TYPE MISMATCHES

Variable type mismatches will stop a data set stacking process in its tracks. A variable type mismatch happens when one data set has a variable that is type numeric and then another data set has a variable that is type character. A program will encounter an error condition and the data will need to be fixed. This is true regardless of the method used to combine the SAS Data Sets.

VARIABLE LENGTH, FORMAT, AND LABEL MISMATCHES

It is possible to get around variable length, format, and label mismatches; but you probably want them to be consistent to avoid data quality and/or value truncation problems. As long as the variable types are the same and the variables have the same name, PROC SQL OUTER UNION CORRESPONDING will use the longest length across all of the data sets while Data Step and PROC APPEND use the first reference to the variable based on the order of the data sets.

WHAT TO DO?

Ask for the source data with the appropriate attribute so we do not have this problem in the first place. Often we do not have control over this and the entity sending the data will not conform. We could spend significant time working with each data source manually and get the attributes consistent. Or we could use our data drive skills along with the SAS Metadata contained in the SAS Dictionary Tables.

A DATA DRIVEN STRATEGY

It is possible to get around variable type, length, format, and label mismatches programmatically by using the SAS metadata. One method uses the SAS Dictionary Tables. The full code will be shared in the presentation but a sample is below. This sample code assumes that the SAS Data Set SASHELP.CLASS has the desired SAS variable attributes and we create NEWCLASSDATA as a hypothetical data set that we want to append to it.

```
*************************************************************************;
** Create a hypothetical data set with different attributes
*************************************************************************;
data newclassdata;
   length name $20 sex $8 height 4 weight 4;
   label name='Student Name';
   format height words60.;
   set sashelp.class(rename=(age=tage));
   age=left(put(tage,2.));
run;

*************************************************************************;
** Read the SAS Data Set Metadata into a SAS Data Set so we can work
** with it programmatically and in a data driven manner.
*************************************************************************;
proc sql;
   create table compare_attributes as
   select target.name,
          target.type   as target_type,
          new.type      as new_type,
          target.length as target_length,
          new.length    as new_length,
          target.format as target_format,
          new.format    as new_format,
          target.label  as target_label,
          new.label     as new_label
   from (select *
         from dictionary.columns
         where libname='SASHELP' and memname='CLASS')    as target,
        (select *
         from dictionary.columns
         where libname='WORK' and memname='NEWCLASSDATA') as new
   where lowcase(target.name)=lowcase(new.name);
quit;
```

Now that the SAS variable attributes are in a SAS Data Set, we can make decisions and generate code in a data driven manner.

Obs	name	target_type	new_type	target_length	new_length	target_format	new_format	target_label	new_label
1	Name	char	char	8	20				Student Name
2	Sex	char	char	1	8				
3	Age	num	char	8	2				
4	Height	num	num	8	4		WORDS60.		
5	Weight	num	num	8	4				

Figure 6. COMPARE_ATTRIBUTES Data Set Resulting from previous PROC SQL.

We can generate code to …

- recognize type mismatches and generate an input or put function
- recognize the maximum length and generate a length statement to ensure data truncation will not take place
- use the target format or generate an appropriate format based on the desired length
- use the target label

CONCLUSION

Using the information and methods in this paper should result in a more stable process for combining SAS data sets. This will improve the consistency and quality of the data you want to use for analysis.

CONTACT INFORMATION

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

Brian Varney
Experis Business Analytics
5220 Lovers Lane, Suite 200
Portage, MI 49002
Phone: 269-553-5185
Fax: 269-553-5101
E-mail: brian.varney@experis.com
Web: www.experis.us/analytics