

DATA Step vs. PROC SQL:

Issues with common variables when combining data sets

By Curt Edmonds, Simulstat Incorporated, and Sunil Gupta, Gupta Programming

In SAS there is more than one way to combine data sets. One method that all SAS programmers are familiar with is the DATA Step. The DATA Step with the merge or set statement is the most common way to combine two or more data sets. But PROC SQL can also be used to combine data sets, sometimes in a more efficient manner. The problem is that the Data Step and PROC SQL differ in the way that the attributes (LENGTH, FORMAT, INFORMAT and LABEL) and values for common non-BY variables are assigned when combining data sets. It is recommended that data sets not contain common non-BY variables to prevent this issue. Common variables in data sets should have the same variable attributes and values.

This tip shows you the general rules on how the attributes and data values from the data sets are assigned for both the DATA Step and PROC SQL when you have common non-BY variables in both data sets.

Below are the sample data sets with four test cases that will be used in the nine examples.

FIRST and SECOND data sets contain four test cases:

1. FIRST data set and SECOND data set have ATTRIB statements for the same variable but with different LENGTH, LABEL and FORMAT– **name**.
2. Default attributes in FIRST data set (LENGTH and FORMAT is \$8, blank LABEL) and ATTRIB statements in SECOND data set for the same variable - **class**.
3. FIRST data set has at least one unique variable with ATTRIB statement – **score**.
4. SECOND data set has at least one unique variable with ATTRIB statement – **grade**.

Note that for testing only, common By variables NAME and CLASS have different lengths. In addition, the CLASS variable in the SECOND data set has data value of length \$10 as compared to the length \$8 in the FIRST data set.

Test Data sets

```
data first;
```

```

    attrib name label='Name from first data set' format=$5.
length=$5
        score label='Score from first data set' format=2.
length=8;

    input name $ class $ score ;
cards;
Tim    math 9
Sally  math 10
John   math 8
;
run;

data second;
    attrib name label='Name from second data set'
format=$10. length=$10
        class label='Class from second data set'
format=$10. length=$10
        grade label='Grade from second data set'
format=$1. length=$1;

    input name $ class $ grade ;
cards;
Tim    mathematic A
Sally  science    C
John   history    B
;
run;

```

Nine examples will be reviewed to observe the results based on various combination methods. The first four use the Data Step and the last five use PROC SQL.

Examples:

1. Merge data sets FIRST and SECOND with BY NAME statement.
2. Merge data sets FIRST and SECOND with BY NAME CLASS statement.
3. Merge data sets FIRST and SECOND without a BY statement.
4. Combine FIRST and SECOND data set with SET statement.

5. Select all variables in FIRST and SECOND data sets using PROC SQL.
6. Select FIRST/LEFT data set variables instead of SECOND/RIGHT data set variables using PROC SQL.
7. Select SECOND/RIGHT data set variables instead of FIRST/LEFT data set variables using PROC SQL.
8. Selecting FIRST/LEFT data set variables or SECOND/RIGHT data set variables using PROC SQL and COALESCE() function.
9. Selecting FIRST/LEFT data set variables or SECOND/RIGHT data set variables using PROC SQL and natural join.

Examples:

1. Merge data sets FIRST and SECOND with BY NAME statement.

The example sorts the data sets by the NAME variable and then merges the data sets with the MERGE and BY statements.

```
proc sort data=first;
  by name;
run;
proc sort data=second;
  by name;
run;

data student;
  merge first
        second;
  by name;
run;

proc print data=student; run;
proc contents data=student; run;
```

As you can see below, the common variables specified in the BY statement have the attributes assigned from the FIRST data set. Note that SAS will issue a WARNING message because the NAME variable has different attributes in the two data sets. It is best to assure common variables have similar attributes. The common variables not specified in the BY statement will have the attributes assigned from the FIRST data set if they are specified but the data will only be from the SECOND data set. This is the case with the CLASS variable where the attributes, other than LENGTH, and values all come

from the second data set. Note that the default attributes of the CLASS variable in the FIRST data set is not considered to be user specified.

NAME values are common in both data sets. (Common BY variable)
CLASS values from second/right data set. (Common non-BY variable)

Obs	name	score	class	grade
1	John	8	history	B
2	Sally	10	science	C
3	Tim	9	mathemat	A

NAME length from first/left data set.
CLASS length from first/left data set.

NAME other attributes from first/left data set.
CLASS other attributes from second/right data set.

#	Variable	Type	Len	Pos	Format	Label
3	class	Char	8	13	\$10.	Class from second data set
4	grade	Char	1	21	\$1.	Grade from second data set
1	name	Char	5	8	\$5.	Name from first data set
2	score	Num	8	0	2.	Score from first data

2. Merge data sets FIRST and SECOND with BY NAME CLASS statement.

The next example better shows how the values are assigned for all the common variables specified in the BY statement. In this example, both the NAME and CLASS variables will be specified in the BY statement.

```
proc sort data=first;
  by name class;
run;
proc sort data=second;
  by name class;
run;

data student;
  merge first
        second;
  by name class;
run;

proc print data=student; run;
```

```
proc contents data=student; run;
```

The common variables specified in the BY statement, NAME and CLASS keep all of the unique values from both data sets. Thus, there are six observations in the data set because there were no matching records between the two data sets. The values for the variables unique to the data sets are kept. The same rules for variables attributes are applied as in example 1.

All records from both data sets since no match between NAME and CLASS.

Obs	name	score	class	grade
1	John	.	history	B
2	John	8	math	
3	Sally	10	math	
4	Sally	.	science	C
5	Tim	9	math	
6	Tim	.	mathemat	A

NAME length from first/left data set. (Common BY variable)

CLASS length from first/left data set. (Common BY variable)

NAME other attributes from first/left data set.

CLASS other attributes from second/right data set.

#	Variable	Type	Len	Pos	Format	Label
3	class	Char	8	13	\$10.	Class from second data set
4	grade	Char	1	21	\$1.	Grade from second data set
1	name	Char	5	8	\$5.	Name from first data set
2	score	Num	8	0	2.	Score from first data

3. Merge data sets FIRST and SECOND without a BY statement.

In the third example, the STUDENT data set is created by using the MERGE statement without a BY statement. In general, this process is not recommended because unrelated data from the two data sets can be linked in the final data set.

```
data student;
  merge first
        second;
run;

proc print data=student; run;
proc contents data=student; run;
```

As you can see below, only the values from the SECOND data set are kept for common variables. This can be seen in the CLASS variable where the values are from the SECOND data set. The same rules for variables attributes are applied as in example 1.

**NAME values are common in both data sets because in same order.
CLASS values from second/right data set.**

Obs	name	score	class	grade
1	John	8	history	B
2	Sally	10	science	C
3	Tim	9	mathemat	A

**NAME length from first/left data set. (Common non-BY variable)
CLASS length from first/left data set. (Common non-BY variable)**

**NAME other attributes from first/left data set.
CLASS other attributes from second/right data set.**

#	Variable	Type	Len	Pos	Format	Label
3	class	Char	8	13	\$10.	Class from second data set
4	grade	Char	1	21	\$1.	Grade from second data set
1	name	Char	5	8	\$5.	Name from first data set
2	score	Num	8	0	2.	Score from first data

4. Combine FIRST and SECOND data set with SET statement.

In this example, the data set STUDENT is created by using the SET statement to combine the FIRST and SECOND data sets. In this example, there is not a BY statement.

```
data student;
  set first
      second;
run;

proc print data=student; run;
proc contents data=student; run;
```

The STUDENT data set has six records (three from FIRST data set, three from SECOND data set) since the two data sets were appended. Data values from both data sets are saved. The same rules for variables attributes are applied as in example 1. Thus, when using the DATA Step to combine data sets with the MERGE or SET statement, SAS applies the same rules when assigning variable attributes.

All records and values from both data sets.

Obs	name	score	class	grade
1	John	8	math	
2	Sally	10	math	
3	Tim	9	math	
4	John	.	history	B
5	Sally	.	science	C
6	Tim	.	mathemat	A

NAME length from first/left data set. (Common non-BY variable)

CLASS length from first/left data set. (Common non-BY variable)

NAME other attributes from first/left data set.

CLASS other attributes from second/right data set.

#	Variable	Type	Len	Pos	Format	Label
3	class	Char	8	13	\$10.	Class from second data set
4	grade	Char	1	21	\$1.	Grade from second data set
1	name	Char	5	8	\$5.	Name from first data set
2	score	Num	8	0	2.	Score from first data

5. Select all variables in FIRST and SECOND data sets using PROC SQL.

The fifth example uses PROC SQL with the SELECT * statement to combine the FIRST and SECOND data sets where the NAME variable values are the same. In this example, since the SELECT * statement is used and there are common variables in the two data sets, SAS will output a WARNING message stating that the common variables already exist in the STUDENT data set.

```
proc sql;
  create table student as
  select *
  from first, second
  where first.name=second.name;
quit;

proc print data=student; run;
proc contents data=student; run;
```

When the SELECT * statement is used, the data and the attributes from the common variables will only come from the FIRST data set. This is evident for the CLASS variable

where all of the values are from the FIRST data set and the label is missing since there was no label created in the FIRST data set. In PROC SQL, the label or other attributes for common variables will not be assigned from the SECOND data set even if they are specified in the SECOND data set. This is an important difference as compared with the DATA Step.

NAME values are common in both data sets. (Common BY variable)
CLASS values from first/left data set. (Common non-BY variable)

Obs	name	score	class	grade
1	John	8	math	B
2	Sally	10	math	C
3	Tim	9	math	A

Length and all other attributes from first/left data set even if first/left data set has default and second/right data set has user specified attributes.
Variables unique to second/right data set have attributes from second/right data set.

#	Variable	Type	Len	Pos	Format	Label
3	class	Char	8	13		(Blank label from first data set)
4	grade	Char	1	21	\$1.	Grade from second data set
1	name	Char	5	8	\$5.	Name from first data set
2	score	Num	8	0	2.	Score from first data

6. Select FIRST/LEFT data set variables instead of SECOND/RIGHT data set variables using PROC SQL.

The next two examples show how the attributes and data values are controlled by using the column alias in the SELECT statement. The column alias allows you to specify exactly which data set (left, right) the variable should come from.

```
proc sql;
  create table student as
  select left.name, left.class, score, grade
  from first as left, second
  where first.name=second.name;
quit;

proc print data=student; run;
proc contents data=student; run;
```

In the output below, you can see that the values from the common variables NAME and CLASS are from the left data set along with their attributes. When the column alias is used in the SELECT statement the values and attributes for common variables will come from the data set specified in the column alias.

NAME values are common in both data sets. (Common BY variable)

CLASS values from first/left data set. (Common non-BY variable)

Obs	name	class	score	grade
1	John	math	8	B
2	Sally	math	10	C
3	Tim	math	9	A

Length and all other attributes from first/left data set even if first/left data set has default and second/right data set has user specified attributes.

Variables unique to second/right data set have attributes from second/right data set.

#	Variable	Type	Len	Pos	Format	Label
2	class	Char	8	13		(Blank label from first data set)
4	grade	Char	1	21	\$1.	Grade from second data set
1	name	Char	5	8	\$5.	Name from first data set
3	score	Num	8	0	2.	Score from first data

7. Select SECOND/RIGHT data set variables instead of FIRST/LEFT data set variables using PROC SQL.

The next example is similar to the previous example except that it shows what happens when the right data set variables are specified in the column alias.

```
proc sql;
  create table student as
  select right.name, right.class, score, grade
  from first, second as right
  where first.name=second.name;
quit;

proc print data=student; run;
proc contents data=student; run;
```

Now that the right data set is specified with the column alias, the values and attributes for the common variables NAME and CLASS come from the right data set.

**NAME values are common in both data sets. (Common BY variable)
 CLASS values from second/right data set. (Common non-BY variable)**

Obs	name	class	score	grade
1	John	history	8	B
2	Sally	science	10	C
3	Tim	mathematic	9	A

**Length and all other attributes from second/right data set even if second/right data set has default and first/left data set has user specified attributes.
 Variables unique to first/left data set have attributes from first/left data set.**

#	Variable	Type	Len	Pos	Format	Label
2	class	Char	10	18	\$10.	Class from second data set
4	grade	Char	1	28	\$1.	Grade from second data set
1	name	Char	10	8	\$10.	Name from second data set
3	score	Num	8	0	2.	Score from first data set

8. Selecting FIRST/LEFT data set variables or SECOND/RIGHT data set variables using PROC SQL and COALESCE() function.

The next example shows how the data values and attributes are assigned when the COALESCE() function is used in the SELECT statement. When the COALESCE() function is used, SAS will check the variables as they are specified in the COALESCE clause and use the data set that has the first non-missing value.

```
proc sql;
  create table student as
  select coalesce(left.name, right.name) as name,
         coalesce(left.class, right.class) as class,
         score, grade
  from first as left, second as right
  where first.name=second.name;
quit;

proc print data=student; run;
proc contents data=student; run;
```

In the output, the variables NAME and CLASS all have the values from the left data set, since they are non-missing. The attributes for these variables will be missing

since the COALESCE clause is creating the new variables without specifying the attributes after the AS keyword. Attributes for these variables can be specified after the variable NAME and CLASS.

NAME values are common in both data sets. (Common BY variable)

CLASS values from first/left data set. (Common non-BY variable)

Obs	name	class	score	grade
1	John	math	8	B
2	Sally	math	10	C
3	Tim	math	9	A

Length of NAME and CLASS is the maximum of two data sets.

Attributes are missing because not specified in COALESCE clause.

Variables unique to each data set have attributes from their corresponding data set.

#	Variable	Type	Len	Pos	Format	Label
2	class	Char	10	18		
4	grade	Char	1	28	\$1.	Grade from second data set
1	name	Char	10	8		
3	score	Num	8	0	2.	Score from first data set

9. Selecting FIRST/LEFT data set variables or SECOND/RIGHT data set variables using PROC SQL and natural join.

In the last example, a natural join is performed where SAS joins the data sets by common variables NAME and CLASS. Only the records that match in both common variables will be kept in the STUDENT data set.

```
proc sql;
  create table student as
  select *
  from first natural join second;
quit;

proc print data=student; run;
proc contents data=student; run;
```

Since the data values for both NAME and CLASS variables are inconsistent between the two data sets, the STUDENT data set has no records. There was no match between any records in the two data sets. In addition, the length of the common variables is the maximum length of the two data sets. The other attributes are missing because these attributes were inconsistent between the two data sets.

No common NAME and CLASS values between data sets – no records created.

```
28 proc sql;
29     create table student as
30     select *
31     from first natural join second;
NOTE: Table WORK.STUDENT created, with 0 rows and 4
columns.

32 quit;
```

Length of NAME and CLASS is the maximum of two data sets. (Common non-BY variables)

Other attributes for NAME and CLASS are missing because not consistent between data sets.

Variables unique to each data set have attributes from their corresponding data set.

#	Variable	Type	Len	Pos	Format	Label
2	class	Char	10	18		
4	grade	Char	1	28	\$1.	Grade from second data set
1	name	Char	10	8		
3	score	Num	8	0	2.	Score from first data set

Based on the nine examples, below are some general key rules applied when combining data sets.

General Key Rules:

- ✓ The LENGTH of common non-BY variables is always determined from the first/left data set whether you are using the MERGE or SET to combine data sets.
- ✓ The LENGTH of common non-BY variables is determined as the maximum value of the first/left data set or the second/right data set when you are using the natural join or PROC SQL COALESCE() function to combine data sets.
- ✓ When using MERGE to combine data sets, common non-BY variable values are kept from the second/right data set.
- ✓ When using PROC SQL to combine data sets, all variable attributes (LENGTH, FORMAT, INFORMAT, LABEL) and data values for common non-WHERE variables are determined and kept from the first/left data set.

Because of the complexity of this issue, the following assumptions are required:

1. Applies when combining two data sets using MERGE or PROC SQL.
2. Two data sets have common variables other than the BY variables.
3. Length and attributes of all BY variables are the same in each data set.
4. Common non-BY variables represent common variables not used in the BY statement.
5. When using the DATA Step to combine data sets, the order of data sets is assumed to be one-to-many. Other configurations such as many-to-one or many-to-many may use the first data set value instead of the second data set value if the second data set has fewer records.
6. Although the table also applies to one-to-one merges, only the match merges are recommended.
7. Variable attributes consist of LENGTH, FORMAT, INFORMAT, and LABEL. Attributes can be specified with the ATTRIB, LENGTH, FORMAT, INFORMAT and LABEL statements.
8. First and left represent the first data set in the merge and PROC SQL FROM statement respectively.
9. Second and right represent the second data set in the merge and PROC SQL FROM statement respectively.
10. When using PROC SQL, warning messages may appear because SAS can not keep the same variables from more than one data set without renaming the variable.
11. When using PROC SQL, common non-WHERE variables is analogous to common non-BY variables in the DATA Step.
12. First data set with or without ATTRIB statements in the prior data step means that the data set was created or was modified with at least one of the following statements: ATTRIB, FORMAT, INFORMAT or LABEL.
13. Second data set, not first data set, with ATTRIB statements in the prior data step means that the data set was created or was modified with at least one of the following statements: ATTRIB, FORMAT, INFORMAT or LABEL.

Source: Sharpening Your SAS Skills, CRC Press (www.sas.com), April 2005

Type	Variable Type	Characteristic	Combination Method	Data Set Contributing	
				First/Left	Second/Right
Data Step	Merge (match merge, one-to-one)	Data Value	All Cases		X ^a
		Length	All Cases	X	
		Format Informat Label	First Data Set with or without Attributes Specified in a Prior Data Step	X	
			Last Data Set, Not First, with Attributes Specified in Prior Data Step		X
	Set	Data Value	All Cases	X	X ^a
		Length	All Cases	X	
		Format Informat Label	First Data Set with or without Attributes Specified in a Prior Data Step	X	
			Last Data Set, Not First, with Attributes Specified in Prior Data Step		X
PROC SQL	Select *	Data Value Length Format Informat Label		X ^b	
	Select <left.var>	Data Value Length Format Informat Label		X ^b	
	Select <right.var>	Data Value Length Format Informat Label			X
	Select Coalesce (<left.var>, <right.var>) as varname	Data Value Length Format Informat Label		X ^c	X ^c

^a = for character variables, data values of shorter length from first/left data set if lengths are unequal

^b = Note, will be missing if not specified in first/left data set and specified in second/right data set

^c = length is the maximum of two data sets, format, informat and label attributes must be specified in COALESCE() clause, due to order specified as <left.var>, <right.var>, data values from second/right data set is used only if data values from first/left data set is missing.

You can find this tip along with other useful techniques in Sunil and Curt's latest book, [*Sharpening Your SAS Skills*](#). Sunil is the principal consultant and trainer at [Gupta Programming](#). Curt is a Managing Partner of [SimulStat Incorporated](#) specializing in SAS programming for the Pharmaceutical Industry.

This book is available from the online bookstore.

[***Sharpening Your SAS Skills***](#)