Overcoming Kainophobia*: Replacing Complex Merges with PROC SQL

Brenda M. Barber, Quintiles, Inc., Research Triangle Park, NC

Abstract

There are many instances in working with data from clinical trials where MERGE is not an adequate tool for joining datasets. Often the data must be extensively manipulated prior to the MERGE, and then un-manipulated afterwards. At times, more than one MERGE must be done to achieve the desired effect, or quite possibly, the dreaded *hard coding* may take place.

PROC SQL has changed all that, and it doesn't have to be difficult to use. Beginning with some basic SQL code, this paper will discuss how to "step up" your code to achieve the truly desired outcome.

Introduction

We all have suffered from *the fear of change at one time or another. Perhaps you have been meaning to investigate the use of PROC SQL, but have been a little shy of what you might be getting yourself into. Maybe the examples in this paper will help you get started. Several aspects of using SQL to combine data files will be explored in this paper, including:

- **joining files when a many-to-many merge is required**
- **many-to-many merges with where-statement exclusion criteria**
- **using sql when working with adverse event and medication dictionaries**
- **combining lab data with normal ranges.**

The examples in this paper refer to pharmaceutical data, but the techniques used will apply to other industries as well.

Combining Lab Data with Normal Ranges:

Sometimes, merging lab data with a file of normal ranges can be tricky. The range values may vary by gender, age of the patient, and the date of the patient's visit. PROC SQL can simplify the task of programming these complicated merges. Let's look at some sample data:

Range File CV_RANGE:

LABTEST	SEX	MIN AGE	MAX AGE	LOW RANG	HI RANG	UNITS
phosphorus	F	0	32	3.2	5.7	mg/dL
phosphorus	F	32	60	2.2	5.1	mg/dL
phosphorus	F	60	999	2.3	5.7	mg/dL
phosphorus	M	0	10	3.4	6.4	mg/dL
phosphorus	M	10	18	3.1	6.0	mg/dL
phosphorus	M	18	999	2.3	5.2	mg/dL
.						
.						
wbc	M	58	999	3.8	10.7	$x10^3$ /uL

For simplicity, no date is shown in this range file, but, at times, the values may vary by date.

Lab Data File BIOCHEM:

PATIENT	VISIT	LABDATE	LABTEST	SEX	AGE	VALUE
001	1	23FEB96	phosphorus		F	32
	3.1					
001	2	02MAR96	phosphorus		F	33
	3.2					
.						
.						
008	1	27FEB96	phosphorus		F	29
	7.0					
.						
.						
059		27APR96	wbc	M	54	11.2

No units are shown on these lab values, again for simplicity, but almost always there would be a units variable present in this file.

Consider the following example code:

```
proc sql;
  create table cv_chem as
  select c.*, r.*
  from biochem as c,file.cv_range as r
  where c.center = r.center
    and c.sex = r.sex
    and c.labtest = r.labtest
    and (r.minage < c.age <= r.maxage);
quit;
```

If you only had to combine these files by gender, the task is easy. Assigning the appropriate ranges by gender and age would be difficult to do with MERGE, and would involve some data manipulation that PROC SQL deems unnecessary.

This code will run, but may terminate with and error-status system message. You will not find "ERROR" in the log, but messages warning that the variables in common to both datasets already exist on file work.cv_chem will be listed. One way to prevent these warning messages from occurring is to specify the variables in each dataset in the SELECT statement, mentioning the variables in common to both files only once.

```
proc sql;
  create table cv_chem as
  select c.center, c.patid, c.sex, c.race,
         c.age, c.visit, c.date, c.labtest,
         c.prot, c.result, c.treat,
         r.low, r.high, r.si_low, r.si_high
  from biochem as c, file.cv_range as r
  where c.center = r.center
    and c.sex = r.sex
    and c.labtest = r.labtest
    and (r.minage < c.age <= r.maxage);
quit;
```

What if a record exists in the lab data, but no ranges are found based on the where statement criteria? The previous code will output only those records that meet the where statement criteria, so WORK.CV_CHEM may contain a fewer number of observations than the lab file BIOCHEM. This is probably not the desired outcome.

The following code will join the lab file with the ranges file, but will keep all the records in the lab file, whether or not any ranges are joined with it.

```
proc sql;
  create table cv_chem as
  select c.center, c.patid, c.sex, c.race,
         c.age, c.visit, c.date, c.labtest,
         c.prot, c.result, c.treat,
         r.low, r.high, r.si_low, r.si_high
```

```
    from biochem as c left join file.cv_range
as r
    on c.center = r.center
        and c.sex = r.sex
        and c.labtest = r.labtest
        and (r.minage < c.age <= r.maxage);
  quit;
```

This is call a LEFT OUTER JOIN and can be done on only 2 datasets at a time. Note that the keyword WHERE is replaced with ON in this type of join.

The two files joined would then look like:

Joined Lab Data File CV_CHEM:

PATIENT	VISIT	LABDATE	LABTEST	SEX	AGE	VALUE
001	1	23FEB96	phosphorus	F	32	3.1
001	2	02MAR96	phosphorus	F	33	3.2
.						
.						
008	1	27FEB96	phosphorus	F	29	7.0
.						
.						
059		27APR96	wbc	M	54	11.2

MINAGE	MAXAGE	LOWRANG	HIRANG	UNITS
0	32	3.2	5.7	mg/dL
32	60	2.2	5.1	mg/dL
.				
.				
0	32	3.2	5.7	mg/dL
.				
.				
58	999	3.8	10.7	x10^3 /uL

Many-to-many Merges

In a clinical trials data file of adverse events (AEs), there may often be many patients who experience the same AE, and they may do so multiple times. The AE may then be coded to more than one preferred term in the dictionary of adverse events, and each preferred term may have several possible body system assignments. SQL provides an easy means to combining the file of AE descriptions with the dictionary. Consider the following:

AE File AES:

PATIENT	VISIT	AEDESC
001	1	hypersalivation
001	2	excessive salivation
001	2	feeling bad
002	1	unsteadiness
002	1	fatigue
002	1	body aches
.		
.		
059	7	excessive salivation

Adverse Events Dictionary File DICTNRY:

AEDESC	PREFTERM	BODYSYS
hypersalivation	saliva increased	autonomic nervous system disorders
hypersalivation	saliva increased	body as a whole - general disorders
escessive salivation	saliva increased	autonomic nervous system disorders
excessive salivation	saliva increased	body as a whole - general disorders
feeling bad	malaise	body as a whole - general disorders
unsteadiness	ataxia	central and peripheral nervous system disorders
unsteadiness	ataxia	body as a whole - general disorders
fatigue	fatigue	body as a whole - general disorders
body aches	malaise	body as a whole - general disorders
drowsiness	somnolence	central and peripheral nervous system disorders

...Etc.

It would be difficult to MERGE these 2 files together without some manipulation to one or both, but joining them using SQL is easy:

```
proc sql;
  create table codedaes as
```

```
        select a.aedesc, a.patient, a.visit,
             d.prefterm, d.bodysys
        from aes as a left join dictnry as d
        on a.aedesc = d.aedesc
        order by patient visit prefterm;
    quit;
```

Note that the variables in common to both files (aedesc) are mentioned only once in the select statement. The LEFT JOIN indicates that all records from the dataset mentioned on the left (AES) will be kept in the output, whether or not they match up with anything in the right-mentioned dataset. The joined file codedaes is displayed on the following page.

It would be possible to add a flag variable to the dictionary file to indicate the default body systems selected for each preferred term in this study. Adding "and d.flag=1" to the ON statement will cause each record in the AES file to join with only one record in the dictionary, so that only one body system is assigned. The number of records output would then equal the number of obs in the original AES file. Here is the codedaes file, showing the hypothetical FLAG variable (only the records with FLAG=1 would be output in that situation):

Joined AE Data File CODEDAES:

PATIENT	VISIT	AEDESC	PREFTERM	BODYSYS	FLAG
001	1	hypersalivation	saliva increased	autonomic nervous system disorders	1
001	1	hypersalivation	saliva increased	body as a whole - general disorders	0
001	2	escessive salivation	saliva increased	autonomic nervous system disorders	1
001	2	excessive salivation	saliva increased	body as a whole - general disorders	0
001	2	feeling bad	malaise	body as a whole - general disorders	1
002	1	unsteadiness	ataxia	central and peripheral nervous system disorders	1
002	1	unsteadiness	ataxia	body as a whole - general disorders	0
002	1	fatigue	fatigue	body as a whole - general disorders	1
002	1	body aches	malaise	body as a whole - general disorders	1
.				.	
059	7	escessive salivation	saliva increased	.	
059	7	excessive salivation	saliva increased	autonomic nervous system disorders	1
				body as a whole - general disorders	0

Similar to the preceding AE file example, it is also difficult to merge concomitant medications with a dictionary file containing multiple Anatomical Therapeutic and Chemical (ATC) Classifications for each drug. Often, the ATC assignments need to be made based on the method by which the drug was administered (route) and/or by the symptom for which the drug was prescribed (indication). The medication file may look something like this:

Medication File MEDS:

PATIENT	VISIT	MED_DESC	ROUTE	INDICATN
001	1	hydrocortisone	topical	skin rash
001	2	hydrocortisone	topical	hemorroids
001	2	lidocaine	topical	hemorroid pain
002	1	hydrocortisone	PO	pre-blood transfusion
002	1	hydrocortisone	IV	chemo premed
002	2	diphenhydramine	topical	

	rash		
		hydrochloride	
003	1	diphenhydramine	topical
	insomnia		
		hydrochloride	
004	1	diphenhydramine	topical
	blood product		
		hydrochloride	premed
001	3	diphenhydramine	topical
	transfusion		
		hydrochloride	reaction

The corresponding medication dictionary file could contain the following:

Medication Dictionary File MDICTNRY:

MED_DESC	ROUTE	ATC_CLAS
hydrocortisone	topical	corticosteroids, weak
hydrocortisone	IV	glucocorticoids
hydrocortisone	topical	products containing corticosteroids
hydrocortisone	topical	corticosteroids for local use
lidocaine	p.o.	anesthetics, local
lidocaine	topical	anesthetics for topical use
lidocaine	IV	amides
diphenhydramine hydrochloride	topical	antihistamines for topical use
dyphenhydramine hydrochloride	IV	aminoalkyl ethers
...Etc.		

Combining these two files to get all of the medications matched up with all of the possible classifications in the dictionary file is very similar to the AE example. If ATC assignments are made, based solely on the route, then this too can be included in the ON statement. However, in the real world, the case report forms are never returned with the route or the indication consistently listed (top, topical, topically,...,PO,P.O., Oral, etc.). In most situations, the meds are joined with all possible classifications, and then a medical reviewer selects the desired classification to be used. If the study is particularly large, or if several similar studies are being done, then possibly a database of defaults based on route and indication could be built. The real world is never as easy as we'd like it to be, but what good is life without a few challenges now and then?!

The SQL code for the medications, then, would look like this:

```
proc sql;
  create table codedmed as
    select m.patient, m.visit, m.med_desc,
           m.route, m.indicatn, d.atc_clas
    from meds as a left join mdictnry as d
    on m.med_desc=d.med_desc
           and m.route=d.route
    order by atc_clas;
quit;
```


Please direct comments or questions to:

Brenda M. Barber
Quintiles, Inc.
Dept. SPS
P.O. Box 13979
RTP, NC 27709-3979