

Merge with SQL

```
PROC SQL;
CREATE TABLE alldata0 AS
SELECT a.*, b.trt_code
FROM adverse a
LEFT JOIN
patdata b
ON a.subject=b.subject;
QUIT;
```

Introduced in SAS 6.07, SQL is a well known language that is very good at working with databases.

Merge in a Data Step

```
DATA alldata0;
MERGE adverse (in=a)
patdata (in=b);
BY subject;
IF a;
```

The most common way data is merged.

Merge with Format

```
DATA fmt;
RETAIN fmtname 'TRT_FMT'
type 'C';
SET patdata;
RENAME subject=start
trt_code=label;
PROC FORMAT CWLIN=fmt;
DATA alldata0;
SET adverse;
ATTRIB trt_code LENGTH=$1
LABEL='Treatment Code';
trt_code=PUT(subject,$trt_fmt.);
```

First load PATDATA into a format, then set TRT_CODE using a PUT statement.

Merge with Modify

```
DATA adverse alldata0;
DO p=1 TO totobs;
iorc =0;
SET patdata point=p nobs=totobs;
DO WHILE (iorc =%sysrc(_sok));
MODIFY adverse KEY=subject;
SELECT (iorc);
WHEN (%sysrc(_sok)) DO; /*Match Found*/
SET patdata POINT=p; OUTPUT alldata0; END;
WHEN (%sysrc(_dsenom)) _error =0; /*No Match*/
OTHERWISE DO; /*A major problem somewhere*/
PUT 'ERR' 'OR: iorc = ' _iorc /
'program halted.'; _error = 0;
STOP;
END;
END;
END;
STOP;
RUN;
```

This is an interesting technique as it is necessary to do a loop within a loop due to the ADVERSE dataset having multiple records per subject.
This method modifies the existing ADVERSE dataset (cannot create TRT_CODE since it will not exist in ADVERSE) and creates a second dataset ALLDATA0. Note also that the dataset ADVERSE has an index called SUBJECT applied before the dataset is run.

Merge with SET-KEY

```
DATA alldata0;
SET adverse;
SET patdata KEY=subject /UNIQUE;
DO;
IF _IORC THEN DO;
ERROR =0;
trt_code='';
END;
END;
```

Many options have been introduced over the years, one of them being the KEY= statement.

Merging Data Eight Different Ways

David Franklin, Independent Consultant, New Hampshire, USA

Introduction
There is more than one way to merge data – this poster looks at eight!

The Data

Dataset:	PATDATA	Dataset:	ADVERSE
SUBJECT	TRT_CODE	SUBJECT	EVENT
124263	A	124263	HEADACHE
124264	A	124266	FEVER
124265	B	124266	NAUSEA
124266	B	124267	FRACTURE

Conclusion
There is no "one method" that is best – it is dependent on the data and the system available to you. But try each one sometime, the results may surprise you.

Merge with an Array

```
DATA null ;
SET sashelp.vtable;
WHERE libname='WORK';
WHERE ALSO memname in ('PATDATA','ADVERSE');
CALL SYMPUT ('X'||memname,put(nobs,8.));
DATA alldata0;
LENGTH trt_code $1; ARRAY f{&xpatdata.,2} $6 _TEMPORARY_;
DO i=1 TO &xpatdata.;
SET patdata (RENAME=(trt_code=trt_code_dict));
f{i,1}=PUT(subject,6.); f{i,2}=trt_code_dict;
END;
DO i=1 TO &xadverse.;
SET adverse;
trt_code='';
DO j=1 TO &xpatdata.;
IF subject=INPUT(f{j,1},best.) THEN DO;
trt_code=f{j,2}; OUTPUT;
END;
IF ^MISSING(trt_code) THEN LEAVE;
END;
IF MISSING(trt_code) THEN OUTPUT;
END;
DROP i j trt_code_dict;
```

A variation on the hash table is to load the dataset with unique records into an array and then do the match. There is one surprising feature - the line: `IF subject=INPUT(f{j,1},best.) THEN DO;` where the actual compare is done, can be changed to use any comparison, whether it be an INDEX function or greater than/less than operators.

Merge with a Hash Table

```
DATA alldata0;
IF _n_ =0 THEN SET patdata;
IF _n_ =1 THEN DO;
DECLARE HASH h1
(dataset="PATDATA");
rc=h1.definekey("SUBJECT");
rc=h1.definidata("TRT_CODE");
rc=h1.definedone();
call missing(SUBJECT,TRT_CODE);
END;
SET adverse;
rc=h1.find();
IF rc^=0 THEN trt_code=" ";
DROP rc;;
RUN;
```

Available since version 9.1, hash tables have been seen as an efficient way to merge data. In the example above, the dataset PATDATA gets loaded into a hash table, then the ADVERSE dataset is loaded into the dataset and the match is made using the FIND() method.

CALL EXECUTE

```
DATA _null_;
SET patdata;
CALL EXECUTE ("DATA alldat;"||
" SET adverse;"||
" WHERE subject="||STRIP(subject)||";"||
" trt_code="||STRIP(trt_code)||";"||
"PROC APPEND BASE=alldata0 DATA=dat0 FORCE;"||
"RUN;");
```

This method uses CALL EXECUTE to add TRT_CODE from PATDATA to ADVERSE by SUBJECT, appending the result each time to the dataset ALLDATA0. Unfortunately this method will only produce a dataset with the intersection of data from PATDATA and ADVERSE, but is something to occasionally use.

Contact Information
David Franklin
Independent SAS Consultant
Tel/Cell: 603-275-6809
Email: dfranklinuk@compuserve.com
http://ourworld.compuserve.com/homepages/dfranklinuk
SGF P197-2009