PharmaSUG2010 - Paper TT15

Utilizing SAS to Dynamically Generate Clinical Trial Data Transformation Programs

Clark Peirce, PPD Inc., Wilmington, NC
Johnny Johnson, PPD Inc., Wilmington, NC

ABSTRACT

We illustrate a technique, using SAS®, to convert programming specification text into a set of uniformly formatted SAS programs used to perform a clinical database transformation. The method begins by reading an Excel worksheet that contains instructions on how to modify clinical database data. Next, we demonstrate how to parse those instructions and convert the text into formal SAS syntax. This syntax is then used to build a series of SAS programs that, when finalized and executed, perform the desired database transformation.

INTRODUCTION

Standardized programming specifications are often used in clinical trial programming to guide the programmer through the data transformation process. These instructions often define the available input and the output expected from the data modification. A typical database transformation consists of modifying variable attributes, converting data types, and creating new variables. Writing the syntax to perform these basic operations is time-consuming for programmers having to manually convert the instructions into executable SAS code.

We illustrate in this paper a process for improving programming efficiency by utilizing SAS to write SAS code. We show how to use the text in a programming specification document to build a set of consistently formatted SAS programs to be used as a starting point in a database transformation. Additionally, we demonstrate how to incorporate custom code, standardized program headers, and other more complex data processing syntax into the programs being created. Figure 1 illustrates a typical data transformation process. Our paper focuses on the SAS Program Creation step in this process.

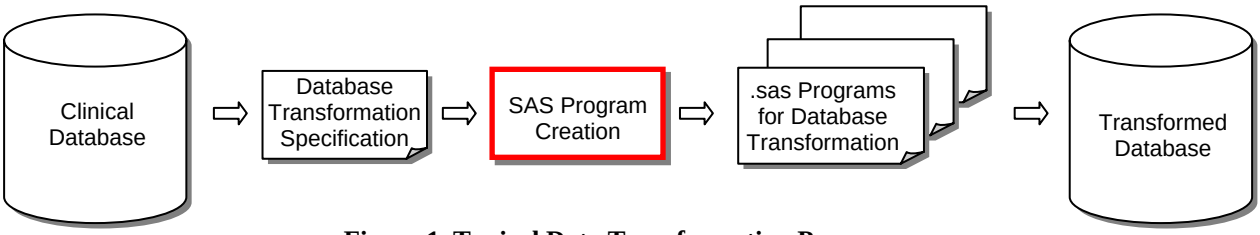

Figure 1. Typical Data Transformation Process

DEFINING A DATABASE TRANSFORMATION SPECIFICATION

The database transformation specification is the primary source of instructions for interpreting and modifying data in the Clinical Database. The data transformation process relies heavily on a well organized specification that contains the following three key elements:

- **Source Metadata Attributes** – source table name, column names, lengths, labels, and data types in the Clinical Database
- **Output Metadata Attributes** – output dataset name, variable names, lengths, labels, and data types in the Transformed Database
- **Derivation Calculations** – specific directions and algorithms for transforming source data to output data.

When these three elements are present, the specification document provides the required information to accomplish typical data transformations such as data type conversions, renaming columns, assigning formats, as well as more complex derivations, e.g. combining multiple character columns to create a single numeric date variable. Most importantly, if the database transformation specifications are consistently formatted across multiple projects then fewer modifications to the example outlined below are required for portability. Programming time can then shift from mundane derivations and program set-up to more complex data algorithms.

Below is a snapshot of the specification used in our paper's data transformation process. It contains separate sets of input and output metadata listed from left to right for the DM and AE clinical database tables needing to be transformed:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**INDATA**	**INNAME**	**INLABEL**	**INTYPE**	**INLENGTH**	**INFORMAT**	**OUTDATA**	**OUTNAME**	**OUTLABEL**	**OUTTYPE**	**OUTLENGTH**	**OUTFORMAT**	**DERIVATION**
2	DM	STUDY	Clinical Study	NUM	8		DEMOG	STUDYID	Study Identifier	CHAR	20	$20.	
3	DM	PT	Patient	CHAR	8		DEMOG	SUBJECT	Subject ID	NUM	8	8.	
4	DM	SEX	Sex	NUM	8	SEXMF.	DEMOG	GENDER	Gender	CHAR	6	$6.	
5	DM	AGE	Age	NUM	8		DEMOG	AGEC	Age	CHAR	3	$3.	
6	DM	BIRTHDT	Birth Date	CHAR	9		DEMOG	BIRTHDTC	Birth Date	CHAR	9	$9.	
7							DEMOG	BIRTHDTN	Birth Date - Numeric	NUM	8	DATE9.	
8													
9	AE	STUDY	Clinical Study	NUM	8		ADVEVE	STUDYID	Study Identifier	CHAR	20	$20.	
10	AE	PT	Patient	CHAR	8		ADVEVE	SUBJECT	Subject ID	NUM	8	8.	
11	AE	AEVERB	Adverse Event VERBATIM	CHAR	200		ADVEVE	AETEXT	Adverse Event Text	CHAR	200	$200.	
12	AE	AESER	Was the AE Serious?	NUM	1	YESNO.	ADVEVE	SERIOUS	Serious Adverse Event	CHAR	3	$3.	
13	AE	AESTDTD	AE Start Date DAY	CHAR	2		ADVEVE	AESTRTD	AE Start Day	CHAR	2	$2.	
14	AE	AESTDTM	AE Start Date MONTH	CHAR	3		ADVEVE	AESTRTM	AE Start Month	CHAR	3	$3.	
15	AE	AESTDTY	AE Start Date YEAR	CHAR	4		ADVEVE	AESTRTY	AE Start Year	CHAR	4	$4.	
16							ADVEVE	AESTDTC	AE Start Date	CHAR	9	$9.	Concatinate AE.AESTDTD, AE.AESTDTM, and AE.AESTDTY
17							ADVEVE	AESTDTN	AE Start Date - Numeric	NUM	8	DATE9.	ADVEVE.AESTDTC

CREATING SAS DATA TRANSFORMATION PROGRAMS

There are four steps to our process:

1. importing the data transformation specification
2. creating variable attribute syntax
3. creating derivation syntax
4. writing SAS syntax to .sas files

STEP 1 – IMPORTING THE DATA TRANSFORMATION SPECIFICATION

We begin by importing the specification Excel spreadsheet using the following SAS code. All header information, blank lines, and rows where there is no variable to derive will be excluded.

```
*******************************************************************************;
* STEP 1 - Read in the mapping specifications;
*******************************************************************************;
PROC IMPORT OUT= WORK.inspec(where = (outdata ^= ''))
            DATAFILE= "C:\INSPEC.XLS"
            DBMS=EXCEL REPLACE;
RUN;
```

Below is a snapshot of the INSPEC dataset generated from this step:

	INDATA	INNAME	INLABEL	INTYPE	INLENGTH	INFORMAT	OUTDATA	OUTNAME	OUTLABEL	OUTTYPE	OUTLENGTH	OUTFORMAT	DERIVATION
1	DM	STUDY	Clinical Study	NUM	8		DEMOG	STUDYID	Study Identifier	CHAR	20	$20.	
2	DM	PT	Patient	CHAR	8		DEMOG	SUBJECT	Subject ID	NUM	8	8.	
3	DM	SEX	Sex	NUM	8	SEXMF.	DEMOG	GENDER	Gender	CHAR	6	$6.	
4	DM	AGE	Age	NUM	8		DEMOG	AGEC	Age	CHAR	3	$3.	
5	DM	BIRTHDT	Birth Date	CHAR	9		DEMOG	BIRTHDTC	Birth Date	CHAR	9	$9.	
6							DEMOG	BIRTHDTN	Birth Date - Numeric	NUM	8	DATE9.	
7	AE	STUDY	Clinical Study	NUM	8		ADVEVE	STUDYID	Study Identifier	CHAR	20	$20.	
8	AE	PT	Patient	CHAR	8		ADVEVE	SUBJECT	Subject ID	NUM	8	8.	
9	AE	AEVERB	Adverse Event VERBATIM	CHAR	200		ADVEVE	AETEXT	Adverse Event Text	CHAR	200	$200.	
10	AE	AESER	Was the AE Serious?	NUM	1	YESNO.	ADVEVE	SERIOUS	Serious Adverse Event	CHAR	3	$3.	
11	AE	AESTDTD	AE Start Date DAY	CHAR	2		ADVEVE	AESTRTD	AE Start Day	CHAR	2	$2.	
12	AE	AESTDTM	AE Start Date MONTH	CHAR	3		ADVEVE	AESTRTM	AE Start Month	CHAR	3	$3.	
13	AE	AESTDTY	AE Start Date YEAR	CHAR	4		ADVEVE	AESTRTY	AE Start Year	CHAR	4	$4.	
14							ADVEVE	AESTDTC	AE Start Date	CHAR	9	$9.	Concatinate AE.AESTDTD, AE.AESTDTM, and AE.AESTDTY
15							ADVEVE	AESTDTN	AE Start Date - Numeric	NUM	8	DATE9.	ADVEVE.AESTDTC

STEP 2 – CREATING ATTRIBUTE SYNTAX

Next we create an ATTRIBUTES dataset that stores all of our derived variable attribute SAS code in a variable called ATTRIBUTE whose values have the following general form:

OUTNAME attribute1 = attribute1 definition … <attribute-n = attribute-n definition>

Where *OUTNAME* is the name of the variable being created and the *attributes* and *attributes definitions* are based on the OUTLABEL, OUTTYPE, and OUTFORMAT columns in the INSPEC dataset. We will use the values of the

ATTRIBUTE variable later to create a single ATTRIB statement.

```
*******************************************************************************************;
* STEP 2 - Create Attribute statements;
*******************************************************************************************;
data attributes(keep = outdata outname attribute);
  length attribute $200;
  set inspec;

  if outname ^= '' then do;
    if outlabel ^= '' then attribute = outname||' label = "'||strip(outlabel)||'"';
    if outtype = 'CHAR' then attribute = strip(attribute)||' length = $'||outlength;
    if outformat ^= '' then attribute = strip(attribute)||' format = '||outformat;
  end;
run;
```

Below is a snapshot of the ATTRIBUTES dataset generated from this step:

	attribute	OUTDATA	OUTNAME
1	STUDYID label = "Study Identifier" length = $20 format = $20.	DEMOG	STUDYID
2	SUBJECT label = "Subject ID" format = 8.	DEMOG	SUBJECT
3	GENDER label = "Gender" length = $6 format = $6.	DEMOG	GENDER
4	AGEC label = "Age" length = $3 format = $3.	DEMOG	AGEC
5	BIRTHDTC label = "Birth Date" length = $9 format = $9.	DEMOG	BIRTHDTC
6	BIRTHDTN label = "Birth Date - Numeric" format = DATE9.	DEMOG	BIRTHDTN
7	STUDYID label = "Study Identifier" length = $20 format = $20.	ADVEVE	STUDYID
8	SUBJECT label = "Subject ID" format = 8.	ADVEVE	SUBJECT
9	AETEXT label = "Adverse Event Text" length = $200 format = $200.	ADVEVE	AETEXT
10	SERIOUS label = "Serious Adverse Event" length = $3 format = $3.	ADVEVE	SERIOUS
11	AESTRTD label = "AE Start Day" length = $2 format = $2.	ADVEVE	AESTRTD
12	AESTRTM label = "AE Start Month" length = $3 format = $3.	ADVEVE	AESTRTM
13	AESTRTY label = "AE Start Year" length = $4 format = $4.	ADVEVE	AESTRTY
14	AESTDTC label = "AE Start Date" length = $9 format = $9.	ADVEVE	AESTDTC
15	AESTDTN label = "AE Start Date - Numeric" format = DATE9.	ADVEVE	AESTDTN

STEP 3 – CREATING DERIVATION SYNTAX

The next step is to create a DERIVATIONS dataset that stores all of the information necessary to create our derived OUTNAME variables in a variable called DERIVE. Using the INSPEC data, the values of DERIVE are based on the DERIVATION variable values as well as the comparison of the INNAME variable attributes with the OUTNAME variable attributes. DERIVE values have the following general form:

OUTNAME = CALCULATION ;

where *OUTNAME* is the name of the variable being created and *CALCULATION* is the derivation expression syntax.

```
*******************************************************************************************;
* STEP 3 - Create derived variable derivation code;
*******************************************************************************************;
data derivations(keep = outdata outname derive where = (derive ^= ''));
  length derive $200;
  set inspec;

  *derivations defined in the specification;
  if derivation ^= '' then derive = '/* '||strip(outname)||' = '||strip(derivation)||' */';

  else do;
    *numeric to character conversions;
    if intype = 'NUM' and outtype = 'CHAR' then do;
      if informat = '' then informat = 'BEST.';
      derive = strip(outname)||' = strip(put('||strip(inname)||','||strip(informat)||'));';
    end;

    *character to numeric conversions;
    else if intype = 'CHAR' and outtype = 'NUM' then do;
      if outformat = '' then outformat = 'BEST.';
      derive = strip(outname)||' = input('||strip(inname)||','||strip(outformat)||');';
```

```
    end;

    *character to character or numeric to numeric assignments;
    else if intype = outtype then derive = strip(outname)||' = '||strip(inname)||';';
  end;

  *customized derivations using a macro call;
  if substr(reverse(strip(outname)),1,3) = 'NTD' then derive =
    '%makedate('||strip(outname)||');';

run;
```

Below is a snapshot of the DERIVATIONS dataset generated from this step:

	derive	OUTDATA	OUTNAME
1	STUDYID = strip(put(STUDY,BEST.));	DEMOG	STUDYID
2	SUBJECT = input(PT,8.);	DEMOG	SUBJECT
3	GENDER = strip(put(SEX,SEXMF.));	DEMOG	GENDER
4	AGEC = strip(put(AGE,BEST.));	DEMOG	AGEC
5	BIRTHDTC = BIRTHDT;	DEMOG	BIRTHDTC
6	%makedate(BIRTHDTN);	DEMOG	BIRTHDTN
7	STUDYID = strip(put(STUDY,BEST.));	ADVEVE	STUDYID
8	SUBJECT = input(PT,8.);	ADVEVE	SUBJECT
9	AETEXT = AEVERB;	ADVEVE	AETEXT
10	SERIOUS = strip(put(AESER,YESNO.));	ADVEVE	SERIOUS
11	AESTRTD = AESTDTD;	ADVEVE	AESTRTD
12	AESTRTM = AESTDTM;	ADVEVE	AESTRTM
13	AESTRTY = AESTDTY;	ADVEVE	AESTRTY
14	/* AESTDTC = Concatinate AE.AESTDTD, AE.AESTDTM, and AE.AESTDTY */	ADVEVE	AESTDTC
15	%makedate(AESTDTN);	ADVEVE	AESTDTN

STEP 4 – WRITING SAS TRANSFORMATION PROGRAM FILES

Now that we have our attribute and derivation syntax datasets, we loop through the INSPEC dataset to generate one SAS program for each dataset being written to the transformed database. The INSPEC, ATTRIBUTES, and DERIVATIONS datasets are used to compose SAS code. We write this code to .sas files using PUT statements. All programs generated through this process will share common syntax formatting, header text, OPTIONS statements, and macro calls, as well as dataset specific ATTRIB statements, KEEP statements, comments, and derivation syntax.

```
*************************************************************************************;
* STEP 4 - Combine the attributes and derivations datasets and create our SAS programs;
*************************************************************************************;
proc sort data = inspec out = alldsn nodupkey;
  by outdata;
run;

%macro mydatasetLoop;
  %local dsid rc nobs varn i value;
  %let dsid = %sysfunc(open(alldsn));
  %let nobs = %sysfunc(attrn(&dsid, nobs));
  %let invar = %sysfunc(varnum(&dsid, indata));
  %let outvar = %sysfunc(varnum(&dsid, outdata));
  %do i = 1 %to &nobs;
    %let rc = %sysfunc(fetchobs(&dsid, &i));
    %let indsn = %sysfunc(getvarc(&dsid, &invar));
    %let outdsn = %sysfunc(getvarc(&dsid, &outvar));
    %let sasfile = "C:\&outdsn..SAS";

    *********************************************************;
    * start by writing out the program header, options,;
    * and any standard %includes;
    *********************************************************;
    proc sort data = inspec (where = (outdata = "&outdsn."))
              out = mapdsn (keep = indata outdata) nodupkey;
      by outdata;
    run;

    data _null_;
      set mapdsn;
```

```
    file &sasfile;
    nowdate = symget('sysdate9');
    put "************************************************************************";
    put "*             CLIENT: CLIENTXXX";
    put "*           PROTOCOL: XXX-XXX-XXX";
    put "*       PROGRAM NAME: &outdsn..SAS";
    put "*        SAS VERSION: 9.1.3";
    put "*            PURPOSE: This program will create the &outdsn. dataset from";
    put "*                     the clinical trial database.";
    put "*        USAGE NOTES: Batch Submit with SAS913.";
    put "*";
    put "*        INPUT FILES: &indsn..SAS7BDAT";
    put "*       OUTPUT FILES: &outdsn..SAS7BDAT";
    put "*";
    put "*             AUTHOR: AUTHORXXX";
    put "*       DATE CREATED: " @;
    put nowdate;
    put "*";
    put "*   MODIFICATION LOG:";
    put "************************************************************************";
    put "* DATE          INITIALS        MODIFICATION DESCRIPTION";
    put "************************************************************************";
    put "*";
    put "************************************************************************;";
    put;
    put "%include ""sassetup.sas"";";
    put "%include ""standardmacros.sas"";";
    put;
    put "options mautosource validvarname = upcase formchar='|_' fmtsearch=(fmtlib.formats);";
    put;
  run;

  ********************************************************;
  * now write out the data step with a keep statement;
  ********************************************************;
  data _null_;
    length txt $256;
    set inspec (where = (outdata = "&outdsn."));
    by outdata;
    file &sasfile mod;
    retain indent;

    if first.outdata then do;
      txt = 'data '||strip(outdata)||' (keep = ';
      put txt @;
      indent = length(txt) + 1;
    end;
    if mod(_n_,8) ^= 0 then do;
      put outname @;
    end;
    else do;
      put outname;
      put +indent @;
    end;
    if last.outdata then do;
      put ');';
      put;
    end;
  run;

  ********************************************************;
  * now write out the attribute statement;
  ********************************************************;
  data _null_;
    length txt $256;
    set attributes (where = (outdata = "&outdsn."));
    by outdata;
    file &sasfile mod;

    if first.outdata then do;
      txt='attrib '||strip(attribute);
      put +2 txt;
    end;
    if not first.outdata then do;
      put +9 attribute;
    end;
    if last.outdata then do;
      put +2 ';';
      put;
    end;
```

```
    run;

    ********************************************************;
    * now write out the derivations for the new variables;
    ********************************************************;
    data _null_;
      set derivations (where = (derive ^= '' and outdata = "&outdsn."));
      by outdata;
      file &sasfile mod;

      if first.outdata then do;
        put +2 "set inlib.&indsn;";
        put;
      end;
      put +2 derive;
      if last.outdata then do;
        put;
        put 'run;';
        put;
      end;
    run;

    ********************************************************;
    * now complete the program by sorting and writing out the;
    * final dataset;
    ********************************************************;
    data _null_;
      file &sasfile mod;
      put "%" "write(dsn = &outdsn, sortby = <ADD &outdsn. SORT VARIABLES HERE>);";
    run;

  %end;
  %let dsid = %sysfunc(close(&dsid));
%mend mydatasetLoop;
%mydatasetLoop;
```

After executing the above SAS syntax, the resulting .sas program files can be used as a starting point or template for further development. The programs can now be examined, updated, and processed independent of one another to complete the database transformation.

Below is the syntax of the ADVEVE.SAS program that is generated from this step:

```
************************************************************************
*             CLIENT: CLIENTXXX
*           PROTOCOL: XXX-XXX-XXX
*       PROGRAM NAME: ADVEVE.SAS
*        SAS VERSION: 9.1.3
*            PURPOSE: This program will create the ADVEVE dataset from
*                     the clinical trial database.
*        USAGE NOTES: Batch Submit with SAS913.
*
*        INPUT FILES: AE.SAS7BDAT
*       OUTPUT FILES: ADVEVE.SAS7BDAT
*
*             AUTHOR: AUTHORXXX
*       DATE CREATED: 03MAR2010
*
*   MODIFICATION LOG:
************************************************************************
* DATE          INITIALS        MODIFICATION DESCRIPTION
************************************************************************
*
************************************************************************;

%include "sassetup.sas";
%include "standardmacros.sas";

options mautosource validvarname = upcase formchar='|_' fmtsearch=(fmtlib.formats);

data ADVEVE (keep = STUDYID SUBJECT AETEXT SERIOUS AESTRTD AESTRTM AESTRTY AESTDTC
                    AESTDTN );

  attrib STUDYID  label = "Study Identifier" length = $20 format = $20.
         SUBJECT  label = "Subject ID" length = $3 format = $3.
         AETEXT   label = "Adverse Event Text" length = $200 format = $200.
         SERIOUS  label = "Serious Adverse Event" length = $3 format = $3.
         AESTRTD  label = "AE Start Day" length = $2 format = $2.
```

```
        AESTRTM  label = "AE Start Month" length = $3 format = $3.
        AESTRTY  label = "AE Start Year" length = $4 format = $4.
        AESTDTC  label = "AE Start Date" length = $9 format = $9.
        AESTDTN  label = "AE Start Date - Numeric" format = DATE9.
  ;

  set inlib.AE;

  STUDYID = strip(put(STUDY,BEST.));
  /* SUBJECT = KEEP LAST 3 DIGITS OF AE.PT */
  AETEXT = AEVERB;
  SERIOUS = strip(put(AESER,YESNO.));
  AESTRTD = AESTDTD;
  AESTRTM = AESTDTM;
  AESTRTY = AESTDTY;
  /* AESTDTC = Concatinate AE.AESTDTD, AE.AESTDTM, and AE.AESTDTY */
  %makedate(AESTDTN);

run;

%write(dsn=ADVEVE,sortby=<ADD ADVEVE SORT VARIABLES HERE>);
```

CONCLUSION

The clinical database transformation process can require relatively simple yet cumbersome programming tasks. Programming specifications are used to document this transformation and to guide the programmer through this process. As an alternative to manually converting these instructions into SAS code, we illustrate in this paper a technique to utilize the specification to dynamically generate SAS programs that form the basis of a database transformation. Automating the SAS program creation saves coding time and enables the programmer to concentrate on more complex programming algorithms.

CONTACT INFORMATION

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

Clark Peirce
PPD, Inc.
929 N. Front St.
Wilmington, NC 28401
Work Phone: 910-558-2413
Fax: 919-654-7462
Email: Clark.Peirce@ppdi.com
Web: www.ppdi.com

Johnny Johnson
PPD, Inc.
929 N. Front St.
Wilmington, NC 28401
Work Phone: 910-558-6020
Fax: 919-654-9988
Email: Johnny.Johnson@ppdi.com
Web: www.ppdi.com