Paper 033-31

Documentation-Driven Techniques for Building SAS® Data Warehouses

Leonid Batkhan, Ph.D., SAS Institute Inc., Cary, NC

ABSTRACT

This paper describes a process that is used in a SAS data warehouse implementation. Data model documentation of a SAS data warehouse (in Microsoft Word or Excel) serves as a feed to a SAS program that performs ETL. Coding techniques are presented for applying SAS formats, informats, and labels to SAS data warehouse tables. It proved to be extremely efficient during development, when data models often undergo changes and modifications. This suggested approach significantly speeds up the implementation of data warehouses.

INTRODUCTION

Building a SAS data warehouse usually involves an Extract-Transform-Load (ETL) process for moving data from databases (DBMS) that are external to SAS such as Oracle, DB2, Informix, Teradata, and so on. SAS/ACCESS software is excellent for extracting those "foreign" tables and converting them into "native" SAS tables. SAS/ACCESS enables you to deal with a DBMS object as if it were a SAS data set.

SAS data sets and DBMS tables have common attributes such as the names, types, and lengths of variables/columns. In addition, SAS data sets contain other variable attributes that do not exist in the DBMS tables such as table labels, variables labels, formats, and informats. To fill that gap, the SAS/ACCESS engine assigns those attributes some plausible values. For example, SAS labels for variables are assigned the values of DBMS table column names, which is not a bad idea given the fact that SAS labels for variables did not exist in the first place.

However, SAS data warehouses are usually built in accordance with data models that contain detailed documentation that specifies all table attributes including labels for variables, formats, and informats. Also, everyone who has ever been involved in a real-life data warehousing project knows that during the development of a data warehouse, such a model goes through multiple changes and modifications.

It is not realistic to try to keep such modifications to a minimum. Usually, the modifications are dictated by the revision of business rules as a project evolves, therefore, it is important to have a single point of recording the modifications in order to eliminate discrepancies between the documentation and the ETL code.

DATA PREPARATION

Suppose that the data model specifications are recorded in a document that is often referred to as the System Design Descriptions (SDD), which usually contains the following:

- A table that lists all the SAS data set names, data set labels, and a string of variable names on which indexes are created.
- For each SAS data set above, a table that lists all variable names, types and lengths, labels, formats, and informats for each data table.

First, you need to capture this information in transportable format files such as CSV (Comma Separated Values) files. If the SDD is an Microsoft Word document, you can copy-and-paste these tables into Microsoft Excel and then save them as CSV files. Let's create the following files: table_list.csv, table1_attr.csv, table2_attr.csv, and so forth. For example,

table_list.csv

tabname	tablabel	tabindex
table1	Table One	var1_2 var1_5 var1_7
table2	Table Two	var2_15 var2_150
table3	Table Three	
table4	Table Four	var4_30 var4_27
……	……	……

table1_attr.csv

varname	vartypelength	varlabel	varformat	varinformat
var1_1	$20	Var One		
var1_2	8	Var Two	datetime20.	datetime20.
var1_3	8	Var Three	mmddyy.	mmddyy.
……		……	……	……

table2_attr.csv

varname	vartypelength	varlabel	varformat	varinformat
var2_1	$80	Var One		
var2_2	8	Var Two	date10.	date10.
var2_3	8	Var Three	datatime20.	datatime20.
……		……	……	……

Now, you can FTP these files to a server that SAS runs on, and convert the files into a set of SAS parameter data sets by using the following SAS code:

```
/* assign a library reference */
libname parmdl '/parmdl';

/* create SAS parameter data set that lists all table names */
data parmdl.table_list;
   length tabname $27 tablabel $40 tabindex $300;
   infile '/table_list.csv';
   input tabname tablabel tabindex;
run;


/* macro definition to loop through all tables */
%macro all_tables_loop;

   %local num_tables i;

   /* determine number of tables */
   data _null_;
      call symput('num_tables',put(n,best.));
      stop;
      set parmdl.table_list nobs=n;
   run;


   %do i=1 %to &num_tables;

        data _null_;
           p=&i;
           set parmdl.table_list point=p;
           call symput('tablename', tabname);
           stop;
        run;

        data parmdl.&tablename._attr;
           length varname $32 vartypelength $6 varlabel $40
                  varformat varinformat $20;
           infile "/&tablename._attr.csv";
           input varname vartypelength varlabel varformat varinformat;
        run;
```

```
      %end;

   %mend all_tables_loop;
   %all_tables_loop;
```

UTILITY MACROS

When you have created SAS parameter data sets that contain metadata information about the tables being created by the ETL process, you can generate the pairs

```
   Variable_name Variable_typelength
   Variable_name = "Variable Label",
   Variable_name  Variable_format,
   Variable_name  Variable_informat,
```

that are needed in the corresponding LENGTH, LABEL, FORMAT, and INFORMAT statements by using the following utility macros:

```
/*
|-------------------------------------------------------------------------------
| NAME        | vlength.sas
|-------------------------------------------------------------------------------
|             |
| DESCRIPTION | is a utility macro that generates the length pairs VARNAME and
|             | VARTYPELENGTH that will be used in a DATA step. For example:
|             |
|             | data tgtlib.table1;
|             |    set srclib.table1;
|             |    length %vlength (parmsl.table1_attr);
|             |       .....
|             | run;
|             |
|             | Input Parameter is the name of the data set that contains the
|             | variables VARNAME and VARTYPELENGTH.
|-------------------------------------------------------------------------------
| SAS VERSION | 9.1.3
|-------------------------------------------------------------------------------
*/
%macro vlength(dataset);

   %local dsid numobs curobs vname vtyplen rc;
   %let dsid = %sysfunc(open(&dataset));
   %if &dsid %then
   %do;

      %let numobs = %sysfunc(attrn(&dsid,NLOBS));

            %do curobs=1 %to &numobs;

                  %let rc = %sysfunc(fetchobs(&dsid,&curobs));
                  %let vname = %sysfunc(getvarc(&dsid,%sysfunc(
                        varnum(&dsid,varname))));
                  %let vtyplen = %sysfunc(getvarc(&dsid,%sysfunc(
                        varnum(&dsid, vartypelength))));

                  &vname &vtyplen

            %end;

            %let rc = %sysfunc(close(&dsid));
```

```
      %end;
   %else %put %sysfunc(sysmsg());

%mend vlength;



/*
|-------------------------------------------------------------------------------
| NAME        | vlabels.sas
|-------------------------------------------------------------------------------
| DESCRIPTION | is a utility macro that generates the pairs VARNAME="VARLABEL",
|             | which will be used in the LABEL statement in a DATA step or in the
|             | DATASETS procedure. For example:
|             |
|             | data tgtlib.table1;
|             |    set srclib.table1;
|             |    label %vlabels(parmsl.table1_attr);
|             |       .....
|             | run;
|             |
|             | or
|             |
|             | proc datasets lib=tgtlib;
|             |    modify table1;
|             |       label %vlables(parmsl.table1_attr);
|             |       .....
|             | quit;
|             |
|             | Input Parameter is the name of the data set that contains the
|             | variables VARNAME and VARLABEL.
|-------------------------------------------------------------------------------
| SAS VERSION | 9.1.3
|-------------------------------------------------------------------------------
*/
%macro vlabels(dataset);

   %local dsid numobs curobs vname vlabel rc;
   %let dsid = %sysfunc(open(&dataset));
   %if &dsid %then
   %do;

      %let numobs = %sysfunc(attrn(&dsid,NLOBS));

             %do curobs=1 %to &numobs;

                    %let rc = %sysfunc(fetchobs(&dsid,&curobs));
                    %let vname = %sysfunc(getvarc(&dsid,%sysfunc(
                           varnum(&dsid,varname))));
                    %let vlabel = %sysfunc(getvarc(&dsid,%sysfunc(
                           varnum(&dsid,varlabel))));

                    &vname = "&vlabel"

             %end;

             %let rc = %sysfunc(close(&dsid));

      %end;
   %else %put %sysfunc(sysmsg());

%mend vlabels;
```

```
/*
|-------------------------------------------------------------------------
| NAME        | vformat.sas
|-------------------------------------------------------------------------
| DESCRIPTION | is a utility macro that generates the format pairs VARNAME and
|             | VARFORMAT, which will be used in a DATA step or in the DATASETS
|             | procedure. For example:
|             |
|             | data tgtlib.table1;
|             |   set srclib.table1;
|             |   format %vformat(parmsl.table1_attr);
|             |        .....
|             | run;
|             |
|             | or
|             |
|             | proc datasets lib=tgtlib;
|             |    modify table1;
|             |       format %vformat(parmsl.table1_attr);
|             |       .....
|             | quit;
|             |
|             | Input Parameter is the name of the data set that contains the
|             | variables VARNAME and VARFORMAT.
|             |
|             | ASSUMPTION: The values of the VARFORMAT variable are suffixed
|             | with a period (.), which is required by SAS syntax and a number, which
|             | is optional.
|-------------------------------------------------------------------------
| SAS VERSION | 9.1.3
|-------------------------------------------------------------------------
*/
%macro vformat(dataset);

   %local dsid numobs curobs vname fname rc;
   %let dsid = %sysfunc(open(&dataset));
   %if &dsid %then
   %do;

      %let numobs = %sysfunc(attrn(&dsid,NLOBS));

             %do curobs=1 %to &numobs;

                    %let rc = %sysfunc(fetchobs(&dsid,&curobs));
                    %let vname = %sysfunc(getvarc(&dsid,%sysfunc(
                           varnum(&dsid,varname))));
                    %let fname = %sysfunc(getvarc(&dsid,%sysfunc(
                           varnum(&dsid,varformat))));

                    &vname &fname

             %end;

             %let rc = %sysfunc(close(&dsid));

       %end;
   %else %put %sysfunc(sysmsg());

%mend vformat;
```

```
/*
|-------------------------------------------------------------------------------
| NAME        | vinformat.sas
|-------------------------------------------------------------------------------
| DESCRIPTION | is a utility macro that generates the informat pairs
|             | VARNAME and VARINFORMAT, which will be used in a DATA step or in
|             | the DATASETS procedure. For example:
|             |
|             |   data tgtlib.table1;
|             |    set srclib.table1;
|             |    informat %vinformat(parmsl.table1_attr);
|             |       .....
|             | run;
|             |
|             | or
|             |
|             | proc datasets lib=tgtlib;
|             |    modify table1;
|             |       informat %vinformat(parmsl.table1_attr);
|             |       .....
|             | quit;
|             |
|             | Input Parameter is the name of the data set that contains the
|             | variables VARNAME and VARINFORMAT.
|             |
|             | ASSUMPTION: The values of the VARINFORMAT variable are suffixed
|             | with a period (.), which is required by SAS syntax, and a number,
|             | which is optional.
|-------------------------------------------------------------------------------
| SAS VERSION | 9.1.3
|-------------------------------------------------------------------------------
*/
%macro vinformat(dataset);

   %local dsid numobs curobs vname ifname rc;
   %let dsid = %sysfunc(open(&dataset));
   %if &dsid %then
   %do;

      %let numobs = %sysfunc(attrn(&dsid,NLOBS));

             %do curobs=1 %to &numobs;

                    %let rc = %sysfunc(fetchobs(&dsid,&curobs));
                    %let vname = %sysfunc(getvarc(&dsid,%sysfunc(
                           varnum(&dsid,varname))));
                    %let ifname = %sysfunc(getvarc(&dsid,%sysfunc(
                           varnum(&dsid,varinformat))));

                    &vname &ifname

             %end;

             %let rc = %sysfunc(close(&dsid));

       %end;
   %else %put %sysfunc(sysmsg());

%mend vinformat;
```

BUILDING A SAS DATA WAREHOUSE BY USING SAS CODE

Using the preceding utility macros, you can build a SAS data warehouse by extracting a source data table, and use SAS code to apply data transformations. For example, if you have an Oracle source data table, you can copy it into a SAS data set and apply the appropriate variable attributes. At the same time, additional transformations can be performed on the SAS data sets using either a DATA step or the DATASETS procedure. For example:

```
Libname tgtlib '/sasdata';
Libname srclib oracle access=readonly user=yyyyy password=xxxxx path=zzzzz;

data tgtlib.table1;
   set srclib.table1;
   label %vlabel(parmsl.table1_attr);
   format %vformat(parmsl.table1_attr);
   informat %vinformat(parmsl.table1_attr);
          .....
run;
```

or

```
data tgtlib.table1;
   set srclib.table1;
.....
run;


proc datasets lib=tgtlib;
   modify table1;
   label %vlabel(parmsl.table1_attr);
   format %vformat(parmsl.table1_attr);
   informat %vinformat(parmsl.table1_attr);
   .....
quit;
```

Transformations can also be applied to all SAS data sets in a macro loop as described in the next section.

BUILDING A DATA WAREHOUSE VIA SAS DATA INTEGRATION STUDIO

Building a SAS data warehouse by using SAS Data Integration Studio (formerly named SAS ETL Studio) involves using interactive plug-ins called Source Designer and Target Designer. These features provide a point-and-click interface for building table metadata and storing it in a metadata repository. The "fun begins" when there are tables with hundreds of variables.

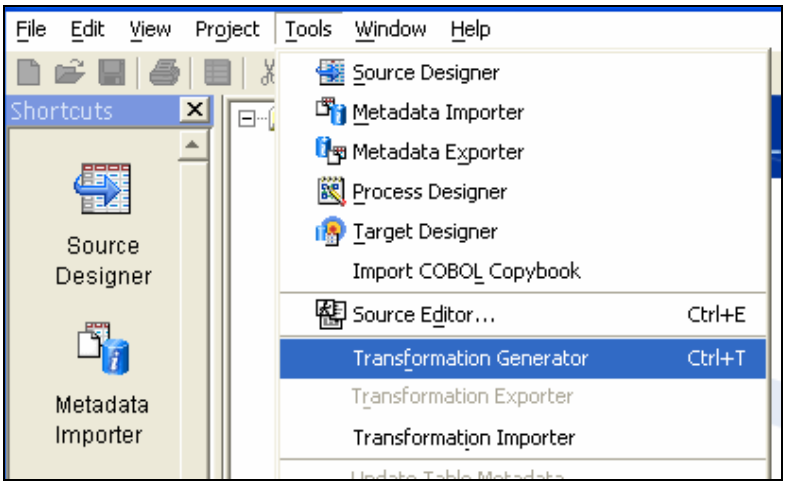
In addition, if the Source Designer works entirely with existing data, the Target Designer usually uses existing tables only partially to build the metadata for the target tables. However, given that design documentation already contains this metadata and that you have already captured it by creating multiple SAS parameter data sets, now you can easily automate the process of building the target tables, thereby supplementing the Target Designer. Suppose, you create the library named "Target Library" in SAS Data Integration Studio or SAS Management Console.

The following code will create all the target tables in the Target Library:

```
/* assign a library reference */
libname parmdl '/parmdl';
libname target meta library='Target Library' repname=Foundation;



/* macro to generate a series of INDEX CREATE statements for PROC DATASETS */
/* parameter indexes is a string of variable names on which to build simple indexes
   */
%macro indexcreate(indexes);
```

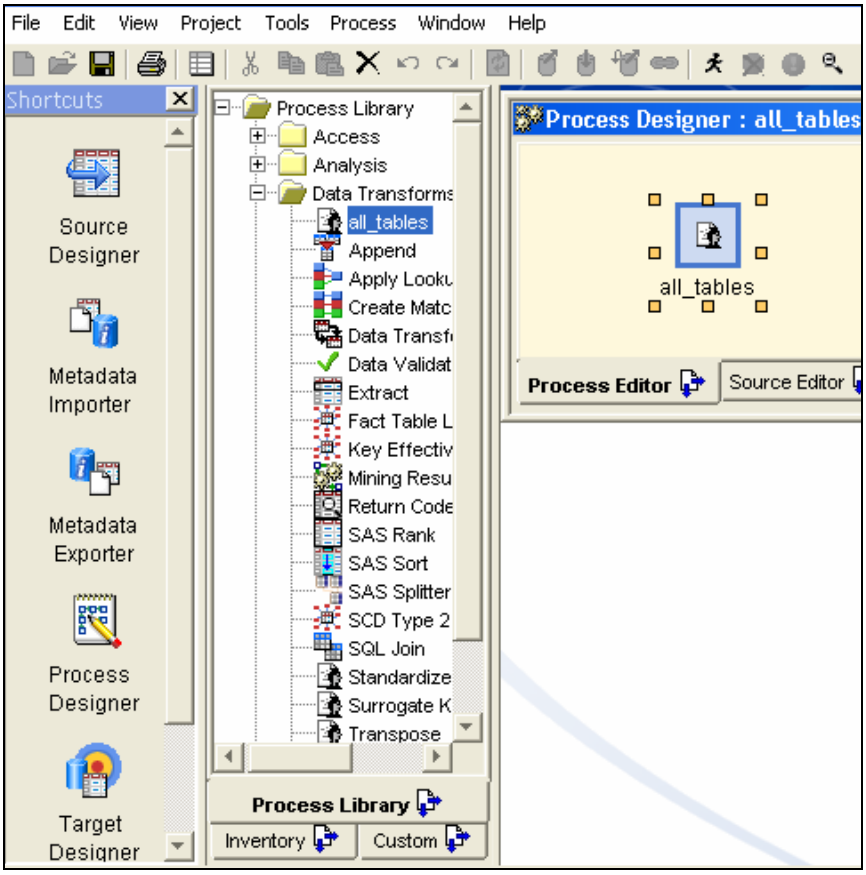
```
   %local var i;
   %let var = %scan(&indexes,1);
   %let i=1;
   %do %while(&var ne );
       index create &var;
       %let i = %eval(&i + 1);
       %let var = %scan(&indexes,&i);
   %end;
%mend indexcreate;


/* macro definition to loop through all tables */
%macro all_tables_loop;

   %local num_tables i tablename tablabel indexstring;

   /* determine number of tables */
   data _null_;
      call symput('num_tables',put(n,best.));
      stop;
      set parmdl.table_list nobs=n;
   run;


   %do i=1 %to &num_tables;

       /* get current table name */
       data _null_;
          p=&i;
          set parmdl.table_list point=p;
          call symput('tablename', tabname);
          call symput('tablabel', tablabel);
          call symput('indexstring',tabindex);
          stop;
       run;

       /* create a zero-observations data set with variable attributes */
    data target.&tablename(label=&tablabel);
      length %vlength(parmsl.&tablename._attr);
      label %vlabel(parmsl.&tablename._attr);
      format %vformat(parmsl.&tablename._attr);
      format %vinformat(parmsl.&tablename._attr);
      stop;
    run;

    proc datasets lib=target nolist;
       modify &tablename;
       %indexcreate(&indexstring);
    quit;

  %end;

%mend all_tables_loop;
%all_tables_loop;
```

This code can be "packaged" as a custom transformation in SAS Data Integration Studio so that it can be re-used by SAS Data Integration Studio users. To do this, open SAS Data Integration Studio and select **Tools**, then select **Transformation Generator** from the resulting drop-down menu (Display 1).



Display 1. Tools Drop-Down Menu in SAS Data Integration Studio

After you give your transformation a name, the Transformation Generator Wizard will open the Source Code window. Paste the preceding code into the Source Code window. Then a job can be created by using the Process Designer (To start the wizard, click the Process Designer icon, which is shown on the left in Display 2.) As you go through the Process Designer Wizard, do not select any tables, just drag-and-drop the preceding transformation from the Process Library onto the Process Editor pane.



Display 2. Process Editor Pane in SAS Data Integration Studio

For more information about using SAS Data Integration Studio, see the "References" section at the end of this paper.

CONCLUSION

This paper presents effective techniques that are applicable to both approaches (that is, using code or using SAS Data Integration Studio) for building SAS data warehouses. These techniques facilitate synchronization of the design documentation that describes the target tables of the data warehouse with the corresponding implementation code or target table definitions in SAS Data Integration Studio. Additional improvement of these techniques can be developed by adopting a design documentation format that SAS can read, and that can be fed directly into the SAS program that re-creates target tables accordingly.

REFERENCES

SAS Institute Inc. 2004. *Instructor-based Training, Course Notes: Using SAS ETL Studio to Integrate Your Data*. Book code 60319. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2006. *SAS® Data Integration Studio 3.3: User's Guide*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGEMENTS

The author thanks Eric Hunley, Scott McCauley, and Stuart Levine for reviewing this paper and providing valuable feedback, and Josephine Pope for being patient and meticulous through the editing process.

CONTACT INFORMATION

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

Leonid Batkhan
SAS Institute Inc.
111 Rockville Pike
Suite 1000
Rockville, MD 20850
Phone: (301) 838-0453, ext. 3307
Leonid.Batkhan@sas.com