

Use of SAS Software to Manage Chemical Plant Laboratory Results

Conrad Reynolds
Exxon Baytown Chemical Plant

Abstract

A large petrochemical plant uses SAS software to manage laboratory sample results. This SAS system includes reports, graphics, long-term storage, and tools to combine the results with process data from non-SAS systems. The paper discusses the system from the programmer's point of view; it covers design, decisions, coding techniques, and system maintenance.

Introduction

Samples of chemical products are regularly collected at the plant, and sent to the plant's laboratory. Each of the samples is identified by a "sample id", and the datetime the sample was collected. The laboratory runs one or more tests on these samples; these tests determine composition or characteristics of the chemical product. The results of the tests are then sent to the SAS system as a set of observations. Each observation is keyed by a concatenated key consisting of the datetime that the sample was taken, a sample identifier, the test run on the sample, and the particular result for that test (some tests produce multiple results).

For example a technician takes a sample of a chemical stream each morning from valve #4 on unit #12. The sample is tagged with a sample id of "UNIT12PT4" as well as the time the sample was taken ("07JUL92:06:30"). This sample is delivered to the lab, where a lab technician runs two tests on it. The first test, called T200, determines the concentrations of benzene and toluene in the sample. The second test, called T250, runs a more sensitive determination of toluene concentration. What is entered into the lab system for this sample looks like this:

```
Sample id:  UNIT12PT4
Date:       07JUL92   Time: 06:30
Test:      T200
Result: BENZENE   Value: .02
Result: TOLUENE   Value: .05
Test:      T250
Result: TOLUENE   Value: .049
```

This information is then sent to the SAS database as the following observations:

SAMPLEID	DATETIME	TEST	RESULT	VALUE
UNIT12PT4	07JUL92:06:30	T200	BENZENE	.02
UNIT12PT4	07JUL92:06:30	T200	TOLUENE	.05
UNIT12PT4	07JUL92:06:30	T250	TOLUENE	.049

The lab data are retrieved by engineers interested in correlating process conditions with the lab analysis of products; however, the process data are not kept in SAS datasets. This paper discusses some of the SAS techniques used to present the data, and some of the outstanding problems with the SAS programs.

Data Storage

The data are stored on a VM/CMS system. The observations are sent from the lab computer to a utility id on the VM system that updates the master datasets using the UPDATE statement. Because of the large number of observations kept (over 100,000 per month), the datasets are kept separately by month. That is, there is one SAS dataset for all January 1992 lab results, one for February 1992, and so forth. This greatly speeds up dataset updating and data retrieval. It does mean that as each month starts, the utility program must create a new SAS dataset for that month, rather than adding observations to the previous month's dataset. I use the handy %EXIST macro from Chapter 11 of the "SAS Guide to Macro Processing" to check if the dataset for that month exists, and if not, the program creates it instead of trying to update it. There are several other handy sample macros in the book, and it is an excellent tutorial and reference to the SAS macro language.

Data Retrieval

Most of the SAS programs that retrieve lab data are written by the engineers themselves. I placed a macro in our SITEAUTO MACLIB autocall library to make coding the SET statements easier. Because the names of the datasets containing current lab data change each month, regularly scheduled jobs could not hard-code SET statements to access the lab datasets. They use a macro ("%SETLAB"), and pass it a parameter of "MTD" for the current calendar month's dataset, or "LMTD" for a concatenation of the last and current month's data.

Many of the engineers need to correlate process data with lab results. Our plants' process data are kept in a Nomad database, keyed by datetime and an instrument id. Each process observation consists of this key and a value. Because SAS does not directly access Nomad databases, a rather complicated macro is used to build a Nomad request, invoke Nomad, and read the returned data into a SAS dataset. This macro is slow and cumbersome, but we improved its performance by "compressing" the SAS source code before storing it in the autocall library. This compressing is accomplished by an XEDIT macro that removes comments, extra spaces, and spaces around special characters. The code is then "flowed" across eighty column lines, and the result looks something like this:

```
DATA &LABVAL;MERGE SNAPNAME SNAPLAB;BY TAGNAME;KEEP TAGNAME SASNAME DATETIME VALUE;%IF
&TRANS=Y%THEN%DO;PROC SORT DATA=&LABVAL;BY DATETIME;PROC TRANSPOSE DATA=&LABVAL
OUT=SNAPTRAN;BY DATETIME;ID SASNAME;IDLABEL TAGNAME;DATA &OUT;SET SNAPTRAN;DROP _NAME_;IF
DATETIME^=.,;%END;%END;%ELSE%DO;
```

This compression cuts about 15% off the execution time for the macro.

Once the engineer has retrieved both the lab data and the process data into SAS datasets, there is still another problem with combining the data. The process data, being regularly scheduled observations, are stored like this:

SAS dataset PROCESS:

<u>DATETIME</u>	<u>TRAY1TEMP</u>	<u>TRAY2TEMP</u>	<u>PIPE1FLOW</u>
12JAN92 13:00	245.7	266.8	33.0
12JAN92 14:00	244.4	266.5	33.2

On the other hand, the lab data are individual events, and are stored like this:

SAS dataset LAB:

<u>DATETIME</u>	<u>SAMPLEID</u>	<u>TEST</u> <u>NAME</u>	<u>RESULT</u> <u>NAME</u>	<u>VALUE</u>
12JAN92 13:00	PIPE33	T222	C3	44
12JAN92 13:00	UNIT32PT4	T345	WATER	.01
12JAN92 13:00	UNIT32PT4	T345	NOAH	11
12JAN92 13:00	UNIT32PT4	T500	NOAH	12
12JAN92 14:00	PIPE33	T222	C3	45
12JAN92 14:00	UNIT32PT4	T345	WATER	.01
12JAN92 14:00	UNIT32PT4	T345	NOAH	11
12JAN92 14:00	UNIT32PT4	T500	NOAH	13

The engineer wants his table to look like this:

SAS dataset COMBO:

<u>DATETIME</u>	<u>TRAY1TEMP</u>	<u>TRAY2TEMP</u>	<u>PIPE33</u> <u>T222</u> <u>C3</u>	<u>UNIT32PT4</u> <u>T345</u> <u>WATER</u>	<u>UNIT32PT4</u> <u>T500</u> <u>NOAH</u>
12JAN92 13:00	245.7	266.8	44	.01	12
12JAN92 14:00	244.4	266.5	45	.01	13

We're going to have to do some sort of transposition here; but what are we going to name the columns labeled "PIPE33 T222 C3", "UNIT32PT4 T345 WATER", and so on? When SAS allows variable name longer than eight characters, then we could just concatenate the values of the sample id, test name, and result name that way. Until then, we need some way of encoding these values into eight characters, and we use the macro facility and PROC SQL to do this. First, we make the simplifying assumption that the engineer will request no more than 99 different sample ids, none of which has more than 99 different tests or results. This allows us to generate variable names of the format "Xaabbcc", where "aa" is a serial number identifying the sample id, "bb" a number for the test name, and "cc" a number for the result name. Since we have thousands of different samples, tests, and results in the entire database, we cannot assign these numbers ahead of time, but instead, must assign them for each engineer's request. To do this, the engineer first subsets the data into a SAS dataset of this form:

Dataset LAB:	<u>variable</u>	<u>format</u>
	DATETIME	DATETIME16.
	SAMPID	\$16.
	TEST	\$4.
	RESNAME	\$6.
	VALUE	\$4.

We create unique variable names for each sample id-test name-result name combination:

```

PROC SQL;
    CREATE TABLE SAMPID AS
        SELECT DISTINCT SAMPID FROM LAB;
    CREATE TABLE TEST AS
        SELECT DISTINCT TEST FROM LAB;
    CREATE TABLE RESNAME AS
        SELECT DISTINCT RESNAME FROM LAB;
/*assign a serial number to each unique value */
DATA SAMPID;
    SET SAMPID;
    XSAMPID = PUT(_N_,Z2.);
DATA TEST;
    SET TEST;
    XTEST = PUT(_N_,Z2.);
DATA RESNAME;
    SET RESNAME;
    XRESNAME = PUT(_N_,Z2.);
/* put the serial numbers back in the original dataset */
PROC SQL;
CREATE TABLE LAB AS
    SELECT SAMPID.XSAMPID, LAB.SAMPID, TEST.XTEST, LAB.TEST,
        RESNAME.XRESNAME, LAB.RESNAME, LAB.DATETIME,
        LAB.RESVALUE
    FROM SAMPID, TEST, RESNAME, LAB
    WHERE SAMPID.SAMPID = LAB.SAMPID AND
        TEST.TEST = LAB.TEST AND
        RESNAME.RESNAME = LAB.RESNAME;
/* create the new variables by concatenating the serial numbers */
CREATE TABLE LAB AS
    SELECT DATETIME, RESVALUE,
        COMPRESS('X'||XSAMPID||XTEST||XRESNAME) AS VARNAME,
        (SAMPID||TEST||RESNAME) AS VARLABEL
    FROM LAB
    ORDER BY DATETIME;

```

Now we can transpose dataset LAB:

```

PROC TRANSPOSE DATA=LAB;
    BY DATETIME;
    ID VARNAME
    IDLABEL VARLABEL;
    VAR RESVALUE;

```

The transposed dataset can now be merged with the dataset PROCESS (above) to achieve the desired results. I've placed the transposing code into an autocall macro for ease of use. For those of you who have not yet discovered the power of PROC SQL, look at the code above, and see how it greatly simplifies the task of combining tables and assigning the new keys.

Engineers who just want to look at the data can run a SAS program that uses a DATA step window to prompt for the calendar month that the engineer is interested in. The program then invokes PROC FSVIEW, the data are presented in "spreadsheet" format; this type of presentation is more quickly grasped and understood by the users than anything else that I have tried.