Accessing Multiple Versions of a Database with One Set of Programs

Spring Tseng, Kirsten MacDonald
Genetics Institute, Cambridge, Massachusetts

ABSTRACT

During a clinical trial, there are times when an interim report needs to be written based on the data available. A snapshot of the complete study database is taken and preserved for report writing. A later version of the on-going database containing more data will be used for another report. Accessing and identifying these two versions of a database co-existing on a system can be a challenging task. Changes to the SAS programs are often needed in order to access the correct database and to identify the database version on the computer output. An efficient way to take care of this situation in UNIX can be accomplished by using two SAS macros and a UNIX shell script. Since all the programs are designed to link to these files, no change is needed once a SAS program is written.

THE SYSTEM

Note: The system described below works in the UNIX environment with data stored in ORACLE. Some knowledge of UNIX and ORACLE will help.

In this article, “database” indicates a collection of all ORACLE tables data pertaining to a protocol.

The two SAS macros in this system are: ‘readdb.sas’ and ‘autoexec.sas’. The UNIX shell script is named ‘runsas’. The UNIX script ‘runsas’ is used to submit a SAS job. Macro ‘autoexec.sas’ is invoked from ‘runsas’. Macro ‘readdb.sas’ is in every SAS program. These files are described to certain detail in this article. A sample program is given at the end and the corresponding output displayed.

I. Macro readdb.sas

This macro reads data from ORACLE to create a SAS dataset.

In an on-going study, the data are changing every minute due to data entry and data cleaning. To reflect the most current data in the data listings and summary tables, data are read in directly from ORACLE at the beginning of each SAS program. This task is done by SAS macro %readdb.sas.

The clinical data are stored in ORACLE tables under database names. The database names are limited to 7 characters with the last character reserved for the database version id. Some valid database names are P96005, P97001a. The protocol name is the same as the database name for an on-going study. The protocol names are limited to 6 characters and never will have a ‘version id’ as a database may. Examples of protocol names are P96005, P97001.

Since the clinical data are collected on an on-going basis, when an interim report is to be written based on the data available, a snapshot of the complete study database is taken. This snapshot of the database is exported for a backup copy. It is imported back to ORACLE for interactive accessing. Upon import, the database name is suffixed with a version id, for example, P97001A. Note while the imported database name is different from the on-going database name, the protocol name remains the same, P97001. This is done at any time during the study, users may look into this version of the database and do further exploration or recreate the previously generated results.

The database snapshots are taken whenever there is a need for an interim report. When the exported database is imported, a version names such as B, C, etc. is assigned as its version id. During the life of the study, there may be several database versions for different purposes. All versions of the database may be active at the same time for the convenience of data checking and report producing.

The database data are broken down to 'crfdata', where 'crf' stands for Case Report Form. Examples of crfdata names are dmg (demographics), sda (study drug administration) and ae (adverse event). In clinical trials, it is common that every database (protocol) has these crfdata.

The database name and crfdata name are required when reading in the data using macro readdb.sas.

The code:

```sas
%macro readdb (database, crfdata);
  proc sql;
  connect to ORACLE (user=&user password=&password path=\ora_path);
  select * from connection to ORACLE (select * from &database &crfdata); 
  disconnect from ORACLE;
  %put %str(NOTE: CRFdata &crfdata has been read for database &database);
%mend readdb;
```

Example 1

```sas
%readdb (p97001, dmg);
```
reads crfdata 'dmg' from the on-going database 'P97001' and creates a temporary SAS dataset 'dmg'.

Example 2

%readdb (p97001a, dmg);

reads crfdata 'dmg' from the imported database 'P97001A' and creates a temporary SAS dataset 'dmg'.

Note example 2 accesses the version 'A' database 'P97001A'. This involves changing the SAS program if the program has been written to access the on-going database. The change can be avoided by using the UNIX shell script 'runsas' described in section III.

II. Macro autoexec.sas

This macro defines the SAS environment parameters and is automatically executed at the beginning of each SAS job.

The code:

%*----------------------------------------------------------;
%* Program : autoexec.sas
%* Purpose :
%* 1. Read arguments from the UNIX shell script 'runsas'.
%* 2. Create SAS macro variables.
%* Notes : Called from 'runsas' shell script.
%*----------------------------------------------------------;
%global progrname database protocol backup user password;

%* convert UNIX arguments into SAS macro variables;
%let progrname = %sysget{program};
%let database = %upcase{%sysget{database}};
%let protocol = %quote{%substr{database,1,6}};

%* The default database version is blank for the on-going database.
%let backup =

%* check if it is an imported database
%if &database ne &protocol %then
%let backup = %substr{protocol,7,1};

%* ORACLE data access keys.
%*----------------------------------------------------------;
%let ora_path = @xxxx;
%let user = xxx;
%let password = xxxxxx;

III. The UNIX Script 'runsas'

A SAS job is submitted through this customized UNIX script.

'runsas' is a UNIX shell script. It requires a minimum of two arguments to run: database (the database name, e.g. p97001, p97001a) and program (the SAS program name, e.g. dmg_001). The code is not included in this article since it is a UNIX shell script.

'Srunsas's Usage

The following are two examples which demonstrate the use of 'runsas' to submit SAS jobs. Note these commands are entered at a UNIX prompt.

Example 1

runsas p97001 dmg_001

This command accesses the on-going database. The arguments p97001 and dmg_001 are UNIX environment variables. They are processed by the 'runsas' script and converted to the values of SAS macro variables &database and &programme.

Example 2

runsas p97001a dmg_001

accesses protocol P97001 version 'A' database, P97001A.

SAMPLE SAS PROGRAM AND OUTPUT

Program name: dmg_001.sas

*----------------------------------------------------------;
* Program : dmg_001.sas
* Purpose : to print patient demographics
*----------------------------------------------------------;
* Note &protocol is created in macro autoexec.sas. &protocol is used as a parameter instead of &database since the value of &protocol remains unchanged for different versions of a database.
%readdb (&protocol, dmg);

data dmg;
set dmg (keepepatno site dob age sex race);

label patno = 'PATIENT NUMBER'
inv = 'SITE NUMBER'
dob = 'DATE OF BIRTH'
age = 'AGE (yr)'
sex = 'SEX'
race = 'RACE';
run;

proc print data=dmg split=' ';
var patno inv dob age sex race;
format dob date. age 3.;
title1 "PROTOCOL: &protocol";
title2 'PATIENT DEMOGRAPHICS';
footnote "&foof';
run;
The output for command: "runsas p97001 dmg_001" looks like:

PROTOCOL: P97001
PATIENT DEMOGRAPHICS

<table>
<thead>
<tr>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMG_001/19APR97</td>
</tr>
</tbody>
</table>

Note the protocol name is printed in the title line, not the database name. The SAS program name and the date the program was run are printed in the footnote.

The output for command: "runsas p97001a dmg_001" looks like:

PROTOCOL: P97001
PATIENT DEMOGRAPHICS

<table>
<thead>
<tr>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMG_001/19APR97/A</td>
</tr>
</tbody>
</table>

Note the protocol name is printed as one of the titles and the database version is printed in the footnote.

When one submits the ‘runsas’ command without specifying the database version number, the ‘runsas’ script accesses the on-going database. No version name will be printed at the output when the current database is used.

CONCLUSION

The system has been put to work for an actual FDA submission. The reports generated using this system include: pre-submission meetings, BLA (Biologic License Application), annual reports and post-submission safety update. It has been proved to be an effective and efficient way of SAS applications development. With two SAS macros in place and a UNIX shell script to submit SAS jobs, the SAS programs written are re-usable for accessing different versions of a database.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. David Eleuteri for his contribution to the system described in this article.

SAS is registered trademark of SAS Institute Inc.
ORACLE is registered trademark of ORACLE Corporation.