Storage Strategies for Data, Formats, Catalogs and Other Information in Application Development Using The SAS® System

Sharon Mosley-Hixon and Ray L Ransom, Centers for Disease Control and Prevention

Abstract

When doing cross-platform development using the SAS System, developers have many options for the storage and access of information needed and generated by their applications. Using the many different engines provided and SAS/ACCESS® software, developers are not restricted to traditional SAS table and catalog storage. Alternative means of storage can enhance an application by minimizing maintenance required, increasing query speeds for large tables, standardizing storage strategies, efficiently using RAM, and utilizing the advantages of the Structured Query Language (SQL). Developers of a sexually transmitted disease (STD) information system (STDINFO) at the Centers for Disease Control and Prevention(CDC) tried storing information in several standard and alternative structures recognizing benefits and short-comings of each approach. This paper will provide a brief description of these strategies and discuss their performance with various SAS/AF® software FRAME classes and certain specific objects. An overview of storage solutions selected for STDINFO will be demonstrated. This information is applicable to developers running SAS version 6.11 on the UNIX, Windows 3.1, Windows NT, and Windows 95 operating systems.

Introduction

STDINFO is an information system being developed at CDC by the Division of Sexually Transmitted Disease Prevention (DSTDP). A decreased programmer to researcher ratio and the increasing computer savvy of the researchers are driving DSTDP programmers into positions of application developers to maximize use of limited programmers' time. FRAME development and true object-oriented programming being relatively new to the SAS system, resources for tips and techniques using SAS Screen Control Language (SCL) are in great demand. Having benefited from many such papers, developers in DSTDP wish to share some of their experiences with other SAS developers.

DSTDP maintains seven surveillance data sets and accompanying format catalogs on an IBM MVS mainframe. This data collection and management process is routine and has remained fairly unchanged for several years. Every year the average researcher and statistician becomes a little more "comfortable" in a "Windows-like" environment due to the ever increasing presence of the PC on desktops and in homes. This trend has resulted in an increased reluctance for scientist to navigate an often cumbersome mainframe environment. As a result, requests for queries on or downloaded subsets of surveillance data are made to the DSTDP programmers. These tasks often take two weeks to complete simply due to heavy work loads. This inefficiency reduces the likelihood of scientists utilizing the surveillance data. Programmers recognize that the relative stableness of these data and the consistency of the requests make application development the obvious solution to this problem. Given that the tables and formats catalogs would be maintained on the network by data management staff, developers proceeded to plan for the application (STDINFO).

Application Requirements

Developers recognized STDINFO must run quickly and be easy to use in order to be accepted by DSTDP staff. To minimize impact on DSTDP resources the application also must require little to no training and minimal maintenance as data are added on a regular basis. The application must be well documented and all processes of preparing tables for the application from the original master files must be automated through batch or cron processing. STDINFO will be used by approximately 150 clients accessing the application from the desktop.

DSTDP Resources

- Master data tables and formats catalogs maintained by data management group
- IBM MVS mainframe
- SUN® UltraSPARC Enterprise 3000 system
- Windows NT Server with SQL server
- Novell application server
- SAS software, version 6.11 wave 2 available on all platforms

- PC Clients ranging from 486/50 16M RAM to Pentium Pro 200 32M RAM
- Windows 3.1/Windows95 on clients
- OLE compliant network software (WP6.1®, Lotus Freelance®, Harvard Graphics®, etc.)

Application Data

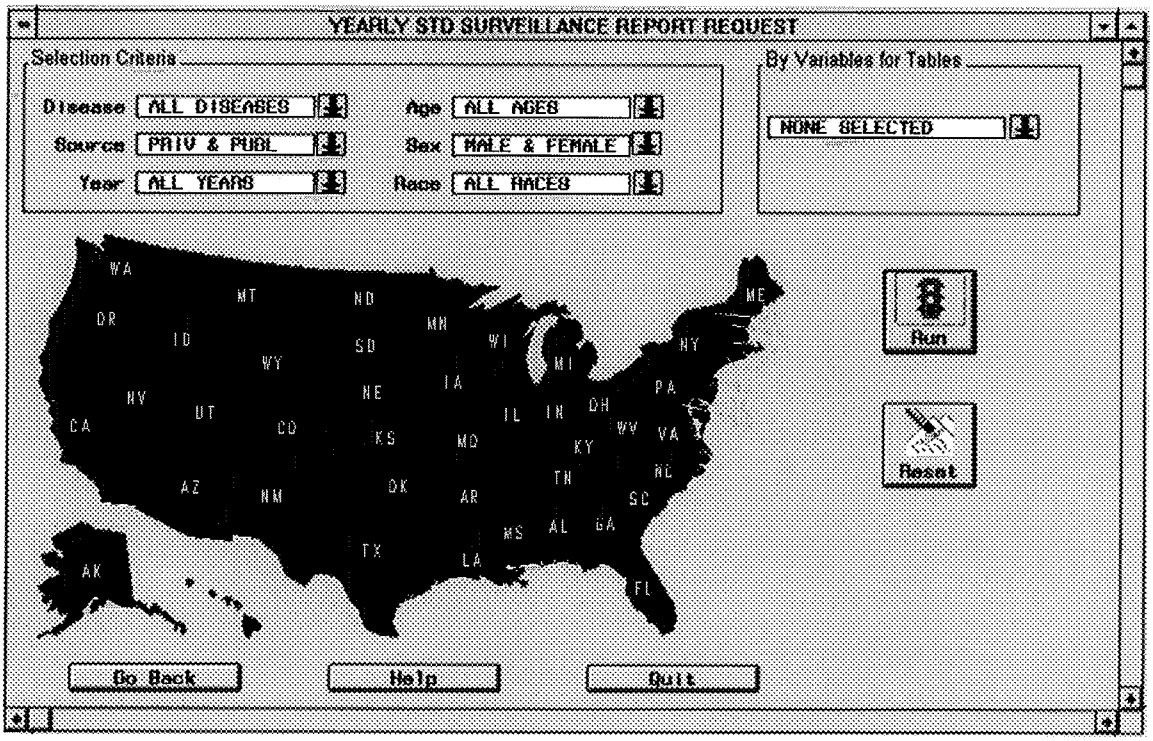
- Indexed table of data to be queried
- Format catalogs
  - -stored value
  - -formatted value
- Variable names
- Variable labels
- Graphics
  - -maps
  - -generated tables/charts
- Frame catalogs and associated SCL code
- Help text/documentation
- Other tables generated in batch for application

Storage Strategies

The master data tables are downloaded from the mainframe and stored on the SUN system. Variable lengths are shortened and an index is built on the key variables used for querying. For further explanation of the systems architecture and the decision to remote submit queries on these indexed tables to the SUN system, please refer to Keith Humphrey's paper in these same proceedings. The primary focus for this paper is the storage of additional information required by STDINFO. At the time of this paper's publishing, the primary objective of the application is to allow users to select subsets of yearly STD surveillance data and calculate rates on selected BY-variables or save subsetted tables for further analysis.

The indexed data table contains basic demographic variables and case and population counts for all strata. The application user is prompted to specify selection criteria and BY-variables using a combination of widgets including a text label, text entry and control object. When the control object is activated an SCL list appears containing all values for that variable. This SCL list somehow must be populated with that information. Once selection criteria are specified and desired states are selected from the hot-spotted map clicking on the RUN icon object remote submits the SQL code including an SQL query to the SUN system (see figure 1 & 2).

Figure 1.



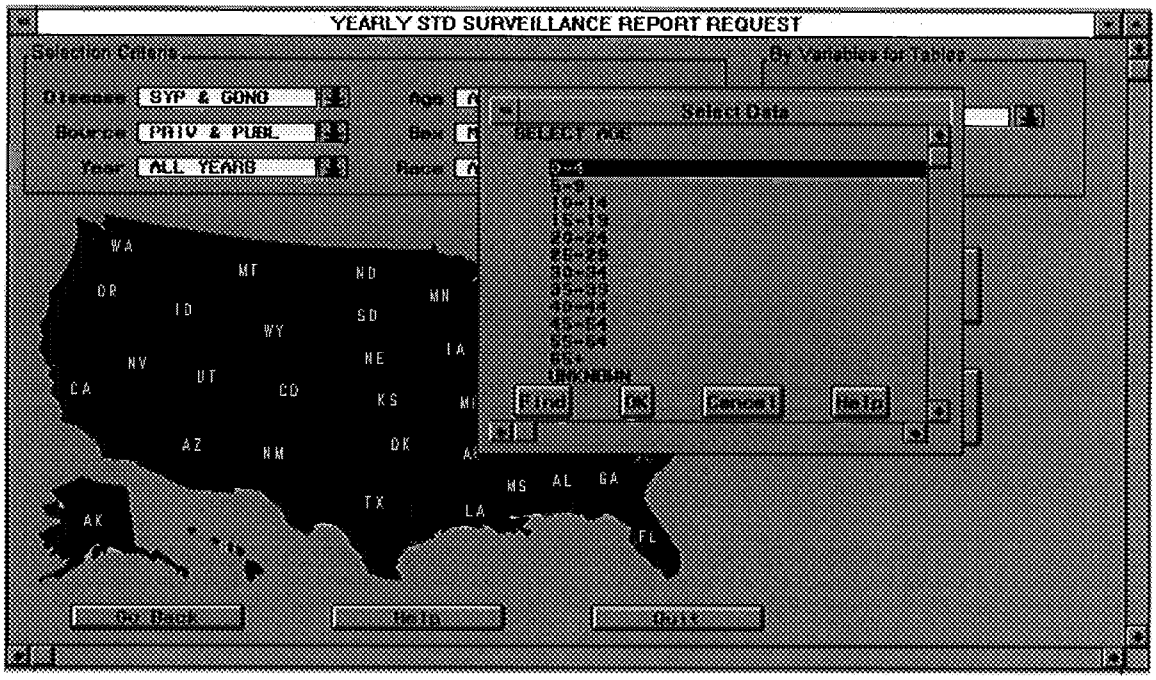


Figure 2.

This one frame requires several pieces of information which can be provided in various ways. For instance, we first tried to use list boxes containing the formatted variable values but soon realized that the screen was not large enough to allow for all the necessary list boxes. Popmenus were the next obvious choice, but this option does not allow for multiple selections, therefore, a control object popping up an SCL list was decided upon.

Now we are faced with how to populate the SCL list. First we populated the list manually in the SCL code behind the frame, but it was pointed out this would require editing the SCL every time new data were introduced to the application (an additional year, for instance) . Given that the data management group would manage the format catalog, we decided to populate the list from the indexed data table using the formatted values. After referencing Don Stanley's paper titled *"Using List To Replace Formats In SCL Applications"* presented at SUGI 21, we decided to create a smaller data set using the CNTLOUT= option in the Frequency Procedure. This eliminated two problems. Populating from the original table was too slow due to the large size of the table. Finally, the inability to control the loading and unloading of formats in RAM was inefficient. Code for generating the formats data set follows.

```
PROC FORMAT
DATA=APP1.APP1DATA
CNTLOUT=APP1.FORMATS;
```

The list is then populate using the following syntax in SCL

```
NAME=DATALIST(LISTID,'YEAR','SELECT YEARS','N',14);
```

Another possible piece of information needed by an application would be the variable label or actual names. Once again, you might not want to get this informatiion from the actual table if the table is large. Similar to how we created a data set from the FORMAT Procedure, we could us the OUT= option in the CONTENTS Procedure to create a small table which could be used to quickly populate an SCL list with syntax similar to the above depending on the characteristics of your object.

The frame displayed above sets all values as the default selection criteria for all variables. This requires a character string to be passed to the SQL selection code within the SCL source code. A string is need for each variable containing all possible values. A data set was created containing these strings using DATA Step processing and the concatenation(||) operator. All of the table creations listed above can easily be run in a batch program that is automatically queued when a change is detected in the master data files.

An especially important piece of information provided to the application by the developers are help files which are displayed using OLE classes from WordPerfect 6.1. This structure allows for easy

creation of help text and creates automatic documentation for STDINFO.

The application itself generates information that must be stored. Although it is rumored to be in a future release in SAS, a DSTDP developer generated a hot-spotted map of the United States for this application. Also each frame is stored in a catalog with SCL source code. It is the decision of the STDINFO development team to store this information and the small tables mentioned above on the local Novell network for quickest access. Should network storage space become a problem or these tables grow considerably in size, Windows NT SQL could be used for storage simply by using SAS/ACCESS. We have successfully accessed tables from the NT server, but current LAN resources make this unnecessary.

Any generated output or tables can be saved to the users hard disk. Subsetted tables may also be saved by the user to their hard disk. By not allowing network storage by the user, we placed the onus of disk space management on the user. A method to determine the most frequently requested tables or graphics by a given application is being developed by DSTDP staff. These images will be stored in a SAS graphics catalog were retrieval time will certainly be much shorter than if they were generated dynamically. Depending on the types these graphics or images, different classes of objects could be used for their display.

Conclusions

Information needed and generated by an application can be stored in a variety of ways thanks to the flexibility of the SAS system. Often times a developers decision is driven by maintenance and performance considerations. STDINFO utilizes what we consider to be smart storage strategy requiring minimal maintenance. What little maintenance that is required is automated. The SCL programming is neither difficult nor sophisticated.

References

SAS Institute Inc. (1989), *SAS Guide to the SQL Procedure, Version 6, First Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1994), *SAS Screen Control Language, Reference, Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

Stanley, Don (1996), "Using Lists To Replace Formats In SCL Applications", *Proceedings of the Twenty-first Annual SAS User's Group International Conference,* 17, 102-107.

Acknowledgments

The authors thank the following people at CDC for their contributions to this paper:
Keith Humphrey and Brenda Sullivan

Sharon Mosley-Hixon
Centers for Disease Control and Prevention, MS E-63
1600 Clifton Road
Atlanta, GA 30333
shm5@cpsstd1.em.cdc.gov

Ray L Ransom
Centers for Disease Control and Prevention, MS E-02
1600 Clifton Road
Atlanta, GA 30333
rlr1@cpsstd1.em.cdc.gov