FRAME Your Mainframe Batch Applications

Michael Davis, Bassett Consulting Services, Inc.

ABSTRACT

SAS® developers often prefer to design applications that run on desktop computers using client-server technology. However, when the applications must read tape data sets too large to warehouse on a disk-based server, some batch processing is usually required. Fortunately, it is unnecessary to forego friendly and flexible graphical user interfaces (GUIs) when portions of an application must be run in batch.

Bassett Consulting Services, Inc. has developed an application for a large financial services client that marries a desktop GUI to sophisticated batch processing. OS/2® and Microsoft® Windows client computers, running the SAS System, write the MVS batch jobs.

Selections are made from graphical screens coded as SAS/AF® FRAME entries. The tape volume (VOLSER) directories and program catalogs are maintained on a file server where they can be shared by client computers. The application consults the VOLSER directory so that only the needed tapes are mounted.

As a result the number of unnecessary tape mounts is reduced by as much as 90 percent. After the batch job is written, it is automatically uploaded and remotely submitted to the host mainframe. The result is an application that even non-programmers find fun to use.

INTRODUCTION:
Welcome to the Real World

Experienced users often observe that there is a considerable difference between using computers depicted by magazines and marketing literature and actual use. Hence, it may be helpful to describe some of the tribulations faced by typical computer users to better appreciate the value of the application to be described. This application is known as Data Builder™.

One popular belief is that mainframe computers are obsolete and will be momentarily supplanted by desktop computers networked to departmental servers running UNIX and other operating systems. The metaphor that often accompanies this belief is instant answers to computer queries supplied by online processing.

While the trend may be accurate, many situations remain where it is not economical to store the massive amounts of information required on magnetic disk drives. In near future, optical storage and other technologies may make it feasible to direct access to nearly all huge databases. However, in 1996, many organizations elect to continue storing their large databases on magnetic tape cartridges.

These cartridges are managed in robotic "silos" and are often connected to large mainframe computer systems, such as those manufactured by IBM Corporation. To increase the productivity of these very expensive resources, most mainframe sites insist that tape cartridges be read through programs submitted for batch-mode processing. Hence, techniques that rely on online access will not work well in these situations.

When computing equipment is shown on television or in a magazine, the increasingly popular view is to display a desktop computer with a graphical user interface (GUI) on the screen. However, as long-time mainframe computer users are aware, mainframe terminal emulators are usually hobbled by the 3270 character-based interface. Also, many 3270 terminal emulators lack support for a pointing device such as a mouse or trackball.

In the ideal environment, computer users have adequate time and training to master their computer tools and their work is interesting and challenging. In the real world, they may have assumed the work that was done by other persons before their organization was "right-sized." Often the opportunity for formal training must be foregone to meet deadlines that seem to arrive ever sooner.

Why SAS ? Client-Server Computing

One reason to use the SAS System in an environment similar to the one that we described is its strong support of client-server computing. In the context of this discussion, we are referring to the SAS System's ability assign each segment of a computer application to the computer platform best able to undertake it. For our purposes, a platform represents any combination of computer hardware and operating systems that are capable of executing the SAS System.

For example, we might start a SAS session on a desktop computer running the SAS system under OS/2 or Microsoft® Windows. Our data resides on tape cartridges stored in a silo connected to a remote mainframe. So we use SAS/CONNECT® to start a second SAS session on the mainframe and transfer a SAS program to be executed as batch submission on the mainframe.

After the mainframe SAS program has executed our remotely submitted program, we might wish to further analyze and graph the extracted information on a departmental server that runs SAS under UNIX. So we again use SAS/CONNECT to transfer the SAS data set from the remote mainframe to departmental server.

Why did we go to this trouble? Each computer was the best choice for the tasks that we directed to it. The mainframe had a direct connection to required tape cartridges and the speed necessary to process the large volume of records to be read from them. Also, mainframe is able to share resources among large numbers of potential users better than the other two platforms.

The desktop computer had an attractive and flexible GUI. Also the marginal costs of an idle desktop computer are considerably less than the other two platforms, making it an ideal choice for use as our often idle "console."

The departmental server represents a good compromise between mainframe and desktop computers when the task load is too high to be quickly processed on a desktop computer. Also, "downsizing" mainframe tasks to departmental servers also saves expensive mainframe cycles.

Why SAS? Portability

Information systems departments frequently ask, "Should our organization stay with our current platforms or switch to others?". An important consideration that ought not be overlooked when considering this question is whether the current application software will run satisfactorily on the new platform. Otherwise, an investment in purchasing or coding new application software will be required. When the known and hidden costs associated with acquiring new application software are summed, they are often largest element of the computing budget.

Bassett Consulting Services develops applications exclusively with the SAS System. A major reason for this policy is because we believe that the SAS System's portability across platforms is unmatched by any other suite of application development tools that we could employ. This protects the investment that our clients make when they retain our services to develop applications for them, should they decide to migrate to another platform.

For example, the client that had commissioned the Data Builder application recently decided that they will migrate from OS/2 to Microsoft Windows. It was a pleasure to tell them that the SAS/AF FRAME entries could be executed on any of the platforms they were considering without modification! Ironically, major portions of the application had been developed under Microsoft Windows Version 3.1 and are maintained under Microsoft Windows 95.

Further, since the SAS catalogs for OS/2, Windows 3.1, Windows 95, and Windows NT share the same SAS catalog format, only one set of application catalogs need be maintained to support users operating Data Builder from any of these desktop platforms. Hence, it is not necessary for all users to be on the same platform to use a single catalog to store the Data Builder application.

This client is also seeking to migrate some of their SAS reporting from a mainframe running MVS to a departmental UNIX server. We are confident that only minor changes to Data Builder will be required to transfer the output files to the server.

One lesson learned during the development of the Data Builder application is that current laptop computers are usually restricted to 800 x 600 pixel resolution. This restriction stems from the LCD panels used for their displays. When we began demonstrating Data Builder to local SAS user groups, problems were encountered displaying FRAME entry screens designed for 1024 x 768 resolution. Since the use of laptop computers as one's primary computer is increasing, we started designing screens to work satisfactorily at 800 x 600 resolution.

Design Goals

When Bassett Consulting Services began developing the Data Builder application, several goals were adopted to guide the development process. We wished to follow sound development practices and to increase client satisfaction with the completed application.

A primary goal was to create an online application that would allow risk analysts to easily extract SAS observations from the raw data stored on tape cartridges and merge the extracts by key values within the same SAS session. The application was to be attractive and easy to use, providing feedback to the user at appropriate points during the screen dialog.

We also sought to make the application so reliable that users could count on the batch jobs submitted by Data Builder running on the first attempt. Besides saving computer resources and preventing user frustration, a reliable application would encourage users to submit jobs for overnight execution, thus conserving mainframe resources during peak daytime hours.

Data Builder was engineered to reduce the number of tape cartridges to be read. In addition to shortening program execution time, this would free up the tape data sets for use by other users sooner, reducing contention.

Another goal that drove the design process was the desire to make the application easy to maintain and change. We sought to reduce the number of changes that would require changing the application's code to as small a set as possible. Last, the client had already created much of the required SAS code. Hence, Data Builder was designed to integrate this code into the submitted jobs and to allow the client to modify these modules without having to recompile the Data Builder FRAME entries.

SCL Lists

After coding an SAS/AF application for another client, we concluded that using CALL DISPLAY to pass parameters among catalog entries was often an unsatisfactory solution. Use of the CALL DISPLAY leaves the calling catalog entries open. If several catalog entries were still open, the closing of the final entry starts a cascade of closing entries that some users found disconcerting. Further, matching the variable lists of the calling entry and called entry was sometimes difficult to coordinate, especially when the order in which catalog entries were opened may vary.

To solve these problems, Bassett Consulting Services began to use SCL lists. SCL lists were added to the SCL (Screen Control Language) with Release 6.07 of the SAS System. Similar in concept to arrays, SCL lists are ordered collections of data. However, SCL lists have additional flexibility that makes them ideal for passing parameters among catalog entries.

Each item of an SCL list can contain a number, character string, or the identifier for another list. A list may simultaneously contain all three types of items. The items can be referenced by their position or by name. SCL lists can be placed in a global environment so that any catalog entry may obtain or modify items. Stored in memory, items in SCL lists can be accessed faster than variables stored in a SAS data set. However, SCL lists can also be stored and retrieved from SLIST catalog entries between SAS sessions.

Because of these features, Bassett Consulting Services decided that the Data Builder application would use SCL lists to replace CALL DISPLAY statements for parameter passing except where only a single FRAME entry window would be opened on top of another. We also decided to aggressively use SCL lists instead of SAS data sets to store all manner of settings and selection lists. By using SCL lists in this manner, we hoped that Data Builder would become more object-oriented in design and be easier to change and maintain.

The next section of this paper contains an abbreviated description of the operation of the Data Builder application. As noted in the abstract, Data Builder is a SAS/AF FRAME application that extracts, merges, and reports information extracted from tape cartridges read by a mainframe computer running the SAS System under the MVS operating system.

The cartridges contain information drawn from approved credit applications, rejected credit applications, and monthly files detailing the account balances and performance. The heart of the Data Builder application is a SAS data set which contains a table listing the clients stored on each tape volume serial number (VOLSER).

Data Builder Main Menu

When a user starts the Data Builder application, a screen similar to the one shown in Figure 1 appears. Please note that all of the screen illustrations (Figure 1 through Figure 6) have been placed at the end of the paper. These particular screens were captured while Data Builder was running under Release 6.11 of the SAS System on a personal computer running Microsoft Windows 95. The user makes most selections either by using a pointing device such as a mouse or by tabbing to a selection and pressing the Enter key.

When the "Build Data Set" icon button is selected, the user is presented with a succession of FRAME entry screens on which are entered the parameters to be used to extract and merge data elements from the tape cartridges. The "Run Reports" icon button brings up a menu of standard reports and allows the user to further subset the data before running a report. The "Maintenance Tasks" icon button brings up the menu shown in Figure 6.

The first FRAME entry screen that appears after the "Build Data Set" icon button is selected is shown in Figure 2. In the box labeled "VOLSERS DIRECTORY DATA AVAILABLE FOR MONTH/YEAR" appears latest date for which applications are available. The range of performance points (the account status "snapshot" as of a particular date) also appears in this box. This information is refreshed by Data Builder at start-up.

The box labeled "CLIENT" selects the customer group menu displayed on the following screen. Breaking the customer list into groups helps to make the selection less intimidating. The box labeled "SAMPLE PERCENT" allows the user to subset the data to be extracted and processed for large customers. The user enters the sample percentage desired in the provided fields.

The box labeled "SETTINGS SOURCE" selects the parameter settings list to be used. Data Builder allows users to retrieve settings, customize them, and save them for reuse. These settings are kept in the global SCL list during a Data Builder session. Between sessions, they are stored as SLIST entries. The use of SCL lists to manage settings helps eliminate the nuisance of referring to notes and cuts repetitive keying.

The "DEPT" (department) list contains those settings values on which the workgroup has standardized. This list is stored on the departmental server. The "LOCAL" list contains the settings preferred by the individual user and is stored on the desktop computer. The "PREVIOUS" list contains the settings used during the last Data Builder session and is saved automatically after each session. The "BUILT-IN" selection is obsolete and not used.

The box labeled "JOB CLASS" selects whether the batch job to extract data from the tape cartridges will be submitted to run as soon as possible ("STANDARD" setting) or deferred for overnight processing. The box labeled "CUSTOM LIST ENTRY NAME" is partial obscured by the SAS status bar. It is used to specify the SLIST catalog entry to be opened when "CUSTOM" is selected from the "CLIENT" box.

Selecting the "CANCEL" push-button returns the user to the Data Builder Main Menu. The "CONTINUE" push-button switches the user to the Data Selection screen after cross-validating screen selections.

Whenever Data Builder detects a problem during cross-validation, a small window pops up. The window title tells the user what problem caused the window to pop-up and a message line in the window suggests how to correct the problem.

Data Selection

The Data Selection screen is shown in Figure 3. Selecting the Core customer list on the previous screen

brings up the Customer Selection List shown in the box labeled "CUSTNO1/CUSTNO2 LIST NAME." Each row of this table represents a sub-list (an SCL list within an SCL list) containing identifiers for each customer.

The client for which Data Builder was designed uses two different identification schemes to associate a credit account with a particular customer. The numbering scheme to be used (Custno1 or Custno2) is selected in the box in the upper-left corner of the screen labeled "SELECT CUSTNO1 OR CUSTNO2 ?"

The Customer Identifier List box in the center of the screen, labeled "CUSTNO1 OR CUSTNO2," is used to view and edit customer identifier SCL lists. The push-buttons to the right of the Customer Selection List provides several functions that work in conjunction with Customer Identifier List box.

The push-button labeled "NEW" clears the Customer Identifier List. The push-button labeled "OPEN" loads the selected customer identifier list into the Customer Identifier List table. If the user prefers, a list can be opened with a mouse by "double-clicking" on the desired list in the Customer Selection List table.

If a customer identifier list is already loaded into the Customer Identifier List table, a pop-up window appears. It offers the user the choice of replacing the existing SCL list with the new one or appending the new list to the end of the existing SCL list. The ability to append lists allows extraction of data for multiple customers through a single extraction run.

Additional rows can be added to the Customer Identifier List by pressing the "ADD" push-button. Customer identifiers are eliminated by deleting the contents of that identifier's row. Blank rows are automatically deleted when the user saves the modified Customer Identifier List to the Customer Selection List by selected the "SAVE" or "SAVE AS" buttons.

Much care was invested in coding the features used to manage the customer identifier lists. Unlike SAS data sets, SCL lists cannot be viewed or edited except through SCL code. An application developer who employs SCL lists bears the added responsibility to provide for any function that a user might need. Thus, the "DELETE" push-button was added to allow a user to remove a row from the Customer Selection List.

To prevent a user from accidentally deleting or corrupting a customer identifier list, Data Builder was engineered to take advantage of security features available in a client-server environment. The SAS catalogs, which contain the FRAME entries and shared SCL lists, and the VOLSER table, are stored on the departmental server. While all users may read from the server, only designated users may also write to the server.

Data Builder retrieves shared information from the departmental server. However, any SCL lists written by Data Builder, such as parameter settings and modified customer identifier lists, are normally written to the disk drive of the desktop computer. When the departmental settings and lists need to be changed, a user with write-privileges to the server performs the update.

The three check boxes within the box labeled "TYPE OF DATA ?" select the type(s) of data to be extracted. When both approved applications and performance records are checked, Data Builder selects only those performance records that correspond to the selected application records and performs a match-merge. Since rejected applications cannot be matched with performance data, rejected applications are appended to the output data set when approved applications or performance records are also selected.

The box labeled "DATE SELECTION" controls the date selection range (month and year) used to filter accounts by the date on which the account was opened. When performance records are requested, the month and year of the desired performance point is also entered.

Extra effort was invested in the coding of the "Date Selection" box to make Data Builder easier to run. While three character month abbreviations are used, the program automatically converts months entered as 1 through 12 to the corresponding abbreviation. Similarly, either the last two digits or all four digits of the year may be entered.

Extensive cross-validation of the selections made on the data selection screen help suppress obvious error that would cause the batch extraction job to fail.

DSN Selection and TSO User ID Entry

Due to space limitations, the next screen is not pictured. On this screen, various data set names (DSNs) are entered. They point Data Builder to SAS code to be included in the submitted batch job. If the feature to which they refer is not to be used, "NULL" is entered instead.

These DSNs point to the scoring models and optional report modules that may need to be included. Because these modules are not part of the compiled catalog entries, the user may alter them at will to accommodate any supplementary processing that might be desired.

Estimated Sample Size

After all of the selection parameters have been entered, Data Builder extracts the observations from the VOLSER table that match the type of data and the customer identifiers selected. This information is used to fill in the "ESTIMATED RECORDS TO BE EXTRACTED" window shown in Figure 4.

The reason why the numbers are shown are estimates is that there is no way to determine in advance how many observations will be filtered by the selected date range. Nevertheless, the information on this screen can help a user determine whether the sample to be extracted may be too large or small, or that no records will be retrieved using the current selections.

If the user decides to try different selection parameters, selecting the "GO BACK" push-button returns Data Builder to the initial selection screen shown in Figure 2. All the current selections are preserved so repetitive entry is minimized. If the user selects the "CONTINUE," the requested program is written to the Preview Buffer Window.

Preview Buffer Window

Data Builder writes a complete MVS batch program based upon the parameters selected and the information extracted from the VOLSER table. The completed program is displayed in browse mode in the Preview Buffer Window. An example of this screen is shown in Figure 5.

The batch program contains a JOB statement, followed by separate job steps for each type of data to be extracted. The extraction is performed by SyncSort™, a third-party sort/merge/copy utility. SyncSort is used for this function because it can extract and filter raw tape records faster than SAS.

The next job step creates SAS data sets from the tape extracts. The Data Builder generated SAS code subsets the SAS data sets by the sampling percentage and re-scores the applications when the external DSNs were included. Last, the processed SAS data sets are merged and/or concatentated into a single output SAS data set.

If the DSN for a report module is entered, an additional job step containing the module is inserted. By including the report as a separate job step, the output data set, which takes the bulk of the execution time to process, is protected in the event of a coding error. When this step fails, it can be rerun by itself.

The user can scroll through the Preview Buffer Window to examine the generated code. To submit the code shown in the Preview Buffer Window, the End command, which is usually assigned to the PF3 key, is issued. This causes the contents of the Preview Buffer to be saved to a file on the desktop computer and the window to be cleared and closed.

If the EHLLAPI communication protocol is used to establish the connection between the desktop computer and the remote mainframe, the user starts up their terminal emulator and signs on to TSO. Data Builder issues the SIGNON command and establishes a link between the two computers through SAS/CONNECT.

Data Builder transfers the batch program from the desktop computer to the remote mainframe. The batch program is submitted for execution by copying the file to the JES internal reader. After the batch program has been submitted, Data Builder issues the SIGNOFF command. After offering the opportunity to save the current parameter settings and to submit another batch

job, Data Builder returns the user to the Data Builder Main Menu.

Maintenance Menu

Figure 6 shows the Data Builder Maintenance Task menu screen. Icon buttons start batch jobs to scan the data cartridges created for each new month or quarter in order to compile the list of the customer identifiers on each cartridge. Other icon buttons cause the compiled lists to be downloaded to the desktop computer. There is also an icon button that deletes obsolete or incorrect data for a performance point as well as one that rolls-back the last update.

As noted earlier, there are no SAS procedures or Display Manager to modify SCL lists. Hence, two of the icon buttons handle SCL update tasks that may be occasionally needed. The "Edit Parameters" icon button changes the name of the catalogued procedure that invokes SAS and the partitioned data set that contains the external SAS code modules. By obtaining these values from a SCL list, the need to edit and recompile catalog entries when these values change is eliminated. The "Copy List" icon button brings up a screen used to copy a selected customer identifier list to another list.

Unlike an application that might be used nearly every day, maintenance tasks are usually performed only weekly or monthly. Sometimes the maintenance tasks are performed by inexperienced persons. If the maintenance tasks are not easy to run, all sorts of problems may occur. Hence, much effort was taken to make maintenance as easy to perform as the extraction and merging of data from the tape cartridges.

CONCLUSION

Even though the Data Builder application has been used by the client for over a year, Bassett Consulting Services is working to further enhance it. Now that a production UNIX departmental server is available, we are adding the ability for Data Builder to transfer the output data sets to it through a screen menu.

An even bigger change is also currently underway. The original version of Data Builder was designed to work with specific files and much of the SAS code is embedded with the FRAME entries. A new version of Data Builder, using the Meta-Master™ library that contains an active data dictionary, is currently under development. When it is completed, users will be able to enter information about raw data sets from maintenance screens.

This version of Data Builder will have the ability to extract and merge information from SAS data sets and views, and SAS/ACCESS® views to relational database systems with information extracted from tape cartridges. Data from any computer that can be reached through SAS/CONNECT can be incorporated into the output SAS data set.

Please direct questions or comments on this paper to:

Michael L. Davis
Vice President
Bassett Consulting Services, Inc.
10 Pleasant Drive
North Haven CT 06473-3712
Telephone: (203) 562-0640
E-Mail: Michael@bassettconsulting.com

ACKNOWLEDGEMENTS

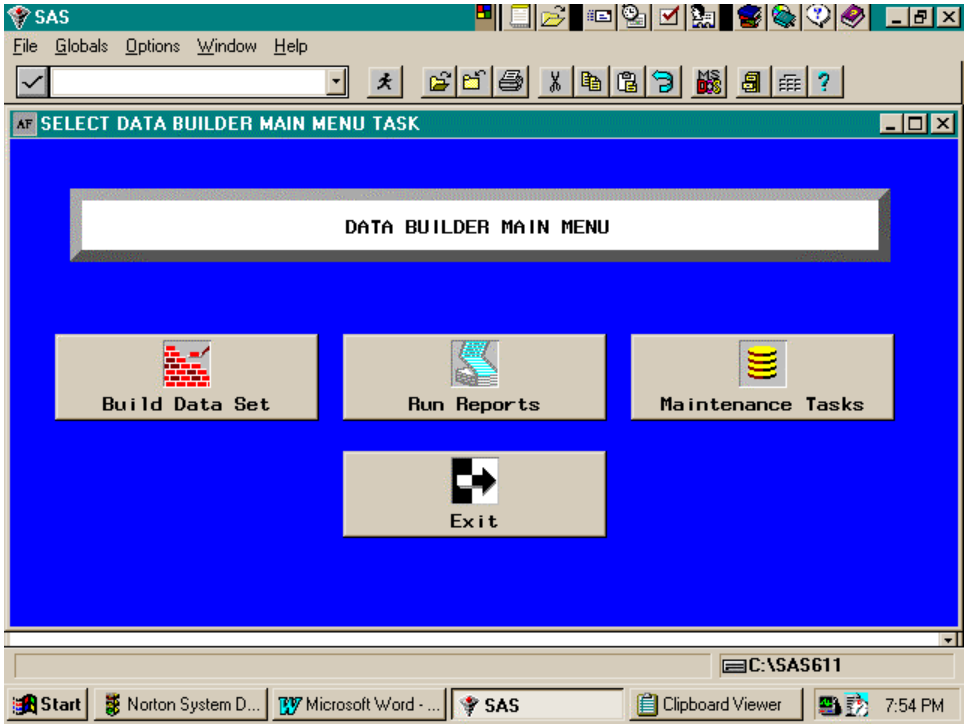

Figure 1

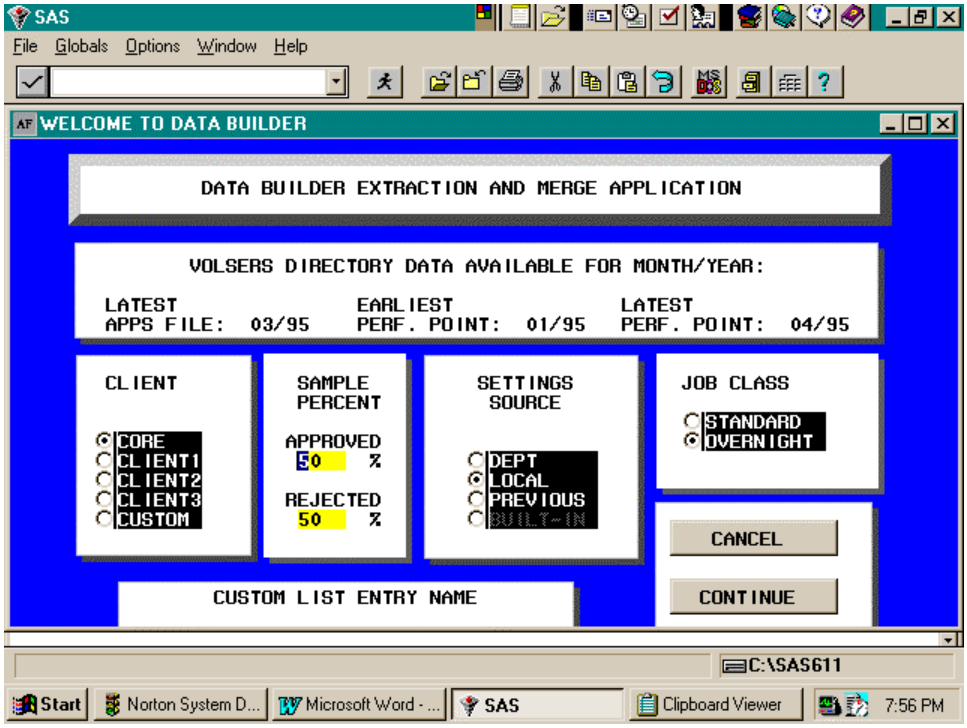

Figure 2

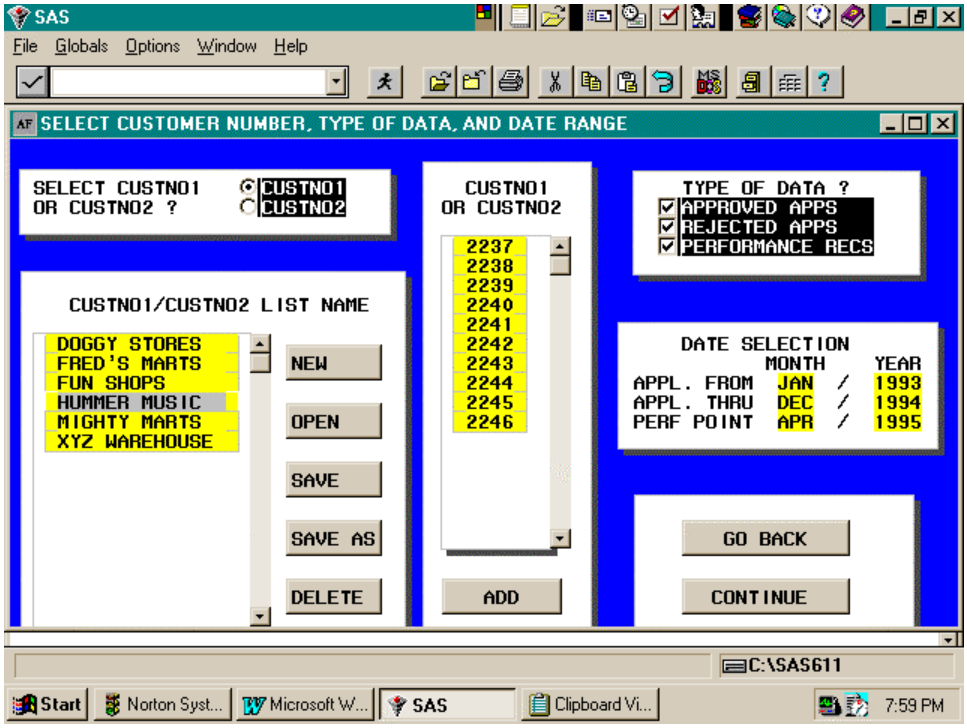

Figure 3

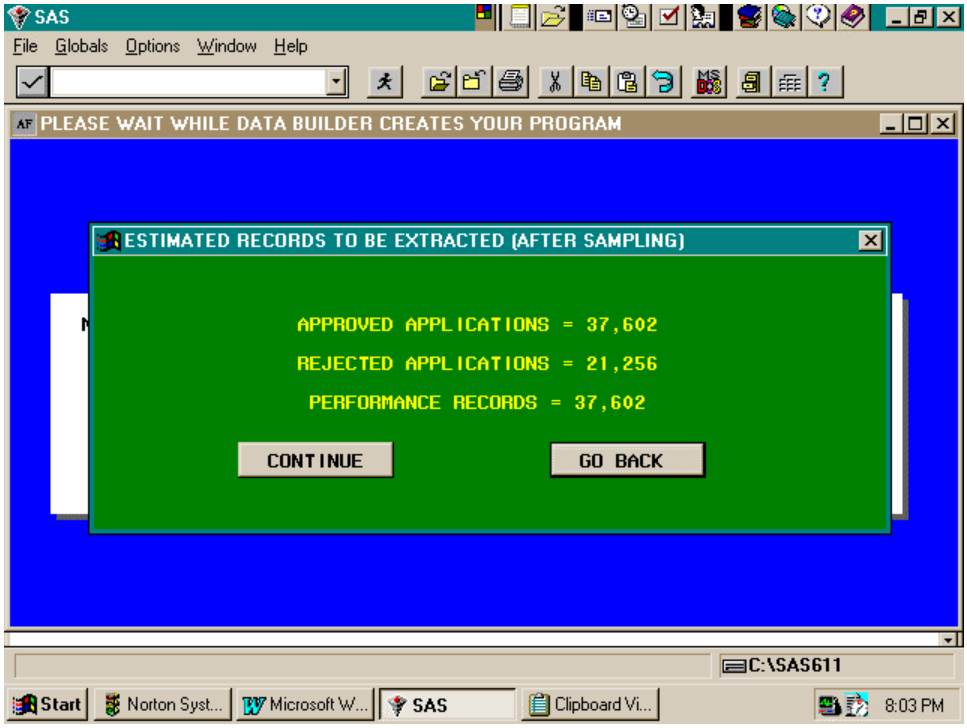

Figure 4

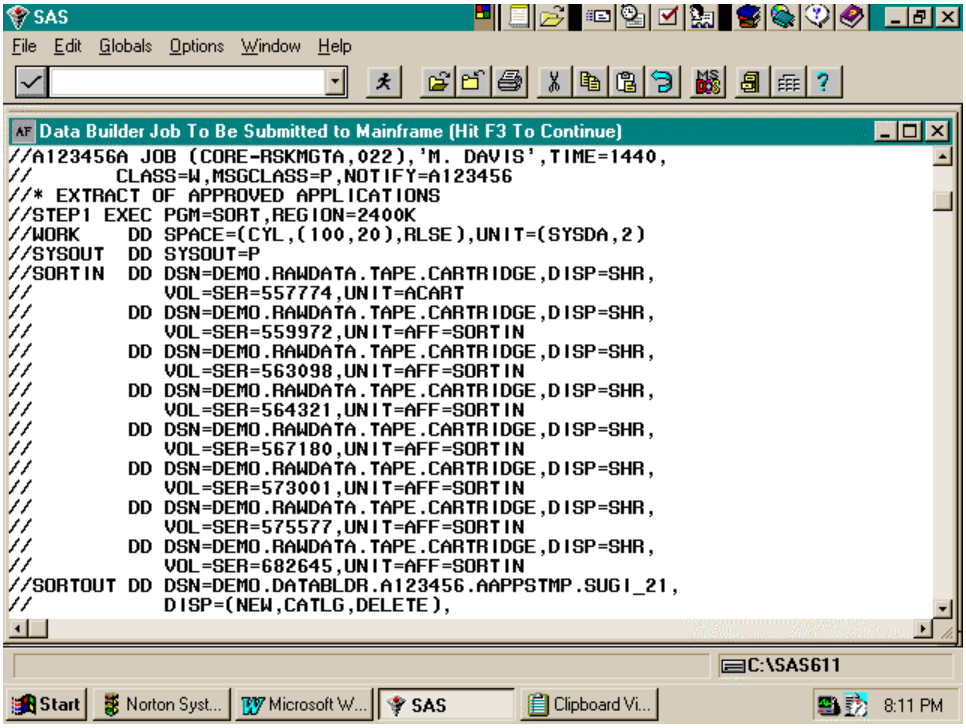

Figure 5

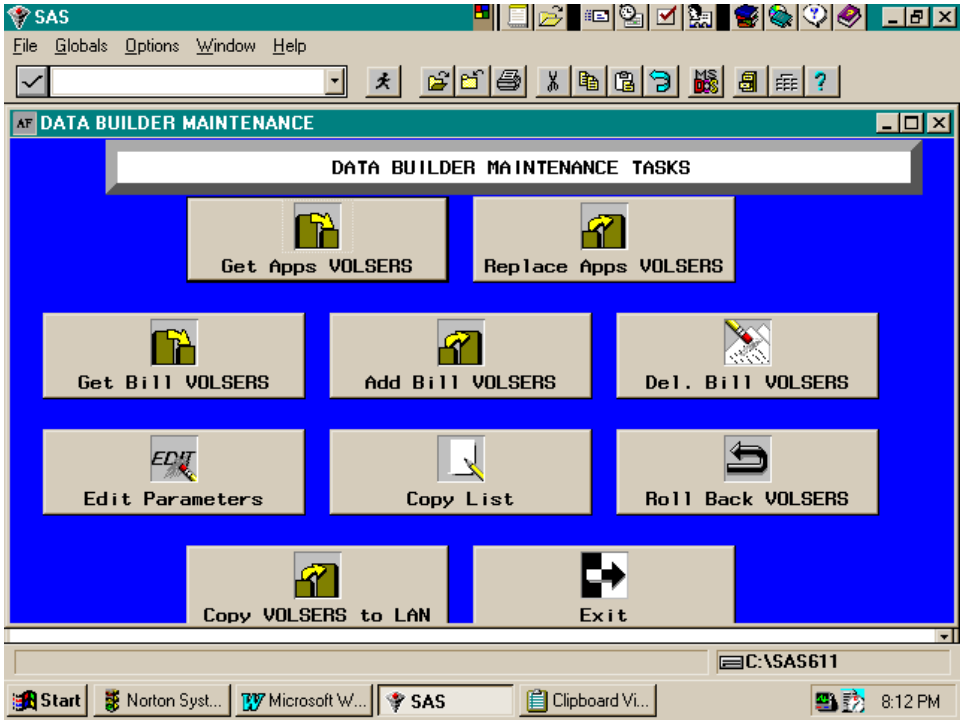

Figure 6