Techniques for Using Screen Control Language in an FSEDIT Multi-user Environment

Ann E. Carpenter, SAS Institute Inc., Cary, NC

ABSTRACT

SAS/SHARE® and SAS/FSP® software work together to allow concurrent update access to SAS® files. Screen Control Language (SCL) applications written with the FSEDIT procedure in a multi-user environment must be aware of locking restrictions imposed by SAS/SHARE software to preserve data integrity. This paper discusses techniques for using SCL in FSEDIT multi-user applications. You should already be familiar with the FSEDIT procedure and Screen Control Language.

INTRODUCTION

SAS/SHARE software allows you to enhance your current SAS/FSP applications by allowing concurrent update access to SAS files. You no longer need to create transaction files and update them to a master data set at night. Now more than one user can access a SAS data file to read or write data at the same time. SAS/SHARE software controls multi-user access by allowing each user update access to one observation at a time.

Concurrent access was available with Version 5 of SAS/SHARE software. But with Version 6 of the SAS System, Screen Control Language (SCL) adds an extra dimension to processing in a multi-user environment.

You may have already written applications that use the FSEDIT procedure and SCL. The same SCL statements and functions apply in a multi-user environment, but now you must be aware of locking considerations on observations you edit and secondary data sets you update. Even if you are only reading a value from a secondary data set that is opened in update mode, if that observation is locked by another user, then you must realize that the value you just read may change.

PROCESSING WITH THE FSEDIT PROCEDURE

To allow multiple people to edit a data set, you must first define your SAS data library to a SAS server. Refer to *SAS/SHARE Software: Usage and Reference, Version 6, First Edition* for information on establishing and accessing a server.

Once you define your SAS data library to a server, you access the FSEDIT procedure as always. For example, define the libref SHARE to your library using a server. Then submit the following statements to invoke the FSEDIT procedure on the data set EMPLOYEE.

```
proc fsedit data=share.employee screen=share.employee;
run;
```

If the library that contains your FSEDIT modified screen is defined to the server, then you are only allowed read access to the screen, not update access. You are also prevented from editing or viewing the SCL program window. If you select option 3 or 6 from the FSEDIT Menu window to access the SCL program, you will receive the following message:

```
WARNING: Program statements are not available. Library is shared.
```

To make modifications to the screen layout or to edit your SCL program, copy the screen entry to a library that is not defined to any server. When the screen is ready for production, you can copy the screen to the library defined to the server.

Access to each observation in the data set is controlled by a SAS server. If another user is currently editing an observation and you scroll to that observation, you will receive a message indicating the observation is in use as shown in Display 1.

```
+FSEDIT SHARE.EMPLOYEE-------------------------------------------------Obs 1+
| Command ===>                                                              |
| WARNING: SASBBL (Server Connection 9) is using this observation.          |
|                                                                           |
|                                                                           |
|  Employee ID Number:          1525                                        |
|                                                                           |
|  Full Name          :  SARAH M. ROBERTS                                   |
|                                                                           |
|  Title              :  SYSTEMS ANALYST              Extension:  3310      |
|                                                                           |
|  Manager            :  GINNY MCALLISTER             Dept code:  DPD       |
|                                                                           |
|  Home Address       :  250 WALKER DRIVE                                   |
|                        CARY, N.C. 27513                                   |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
|                                                                           |
+---------------------------------------------------------------------------+
```

Display 1

When you scroll to an observation that is currently in use, your server prevents you from editing that observation. This record-level locking assures the user who is updating the observation that as long as he is on that observation, all of his changes will be saved.

To gain update access to an observation in an FSEDIT session, you can either read the same observation again or scroll to another observation. Issue the REREAD command on the command line of the locked observation to determine if that observation is still being updated by someone else. If the other user has finished his updates and left the observation, you may now be allowed to update the observation yourself. If this observation is still locked, you can scroll to another observation or issue any of the FSEDIT searching commands to locate another observation. As that observation is displayed, you will again receive a warning message if it also is locked, and you can continue until you find an observation that no one else is currently editing.

When searching for a value in an FSEDIT multi-user environment, it is more efficient to issue the WHERE command than any of the other searching commands such as FIND, LOCATE, or SEARCH. A WHERE clause is evaluated in the server and only the matching subset is returned to the user. This reduces the amount of data communicated between the server and the user's SAS session and the processing time required to evaluate the search condition. A WHERE clause may also take advantage of indexes to make searching on indexed variables even faster.

FULL-SCREEN PROCEDURES THAT LOCK OBSERVATIONS

Now that you are familiar with how observations can be locked by other users running FSEDIT sessions on the same data set, you must consider other ways observations can be locked as well. Observations from the data set you specify on your FSEDIT invocation are typically locked by:

- other FSEDIT sessions
- FSVIEW sessions.

An FSVIEW session opened in edit mode with record-level access displays a table of observations but still only allows you to edit the data one observation at a time. You move your cursor to the desired observation and press ENTER. At this point the server displays a message if the observation is already in use, shows any updated values in the observation, and prevents you from making changes to the existing values.

You cannot issue the REREAD command as in the FSEDIT procedure, but to try again to gain access to a particular observation, move your cursor back to that observation and press ENTER. If you can now edit the observation, its values display in reverse video.

SCL CONSIDERATIONS

In writing your FSEDIT SCL application, you must be aware that you can scroll to an observation that could be locked by another FSEDIT or FSVIEW session. If your SCL code has CONTROL ENTER or ALWAYS specified, statements in the MAIN section will execute when you press ENTER, even though the observation is locked. If your SCL program requires you to enter a value before leaving the observation, yet the observation is locked, you have coded yourself into a corner.

The OBSINFO SCL function allows you to prevent this situation. OBSINFO is only available in the FSEDIT procedure and provides information on the current observation. You can use OBSINFO to query whether the observation is locked and then conditionally decide which parts of your SCL program should execute.

For example, if you don't want any statements in the MAIN section to execute when the current observation is locked, add the following SCL statement at the beginning of the MAIN section:

```
if obsinfo('locked') then return;
```

Here, if you scroll to an observation that is locked, the condition is met and no other statements in this section execute.

SCL FUNCTIONS THAT LOCK OBSERVATIONS

Now that you have dealt with editing observations that could be locked by other full-screen procedures, you must be aware of observations in secondary data sets that could also be locked. Many FSEDIT applications open secondary data sets in update mode to read and update values. If the observation you read is locked, you must realize that someone else could be making changes to the values in that observation. If you attempt to update an observation that is locked, your update will be unsuccessful.

SCL functions that fetch an observation into the Data Set Data Vector (DDV) and lock that observation include the following:

- FETCH
- FETCHOBS
- LOCATEC and LOCATEN
- DATALISTC and DATALISTN.

Once your SCL program reads an observation with one of these functions, the observation is locked until the program reads another observation or executes the UNLOCK function to release the lock on the current observation.

If you are fetching a value from an observation and don't intend to change the values in that observation, then issue the UNLOCK function immediately after reading the observation. This limits the time the observation is locked to another user who may want to update it.

The FETCH Function

The function most commonly used in a multi-user environment to lock an observation is the FETCH function. FETCH is typically used in conjunction with the WHERE function. As with the WHERE command, the WHERE function is the most efficient means to subset a secondary data set defined to a server. The WHERE clause is evaluated in the server thus reducing processing time required to evaluate the search condition. The WHERE clause may also take advantage of indexes.

Once the WHERE function subsets the data set, the FETCH function reads in the first observation that meets the WHERE condition. FETCH attempts to gain a lock on the observation being read. If the observation is already locked by another user, FETCH still reads the observation and indicates that the observation is locked.

For example, the following code fragment applies a WHERE clause to the data set referenced by DSID where SSN is the variable on the data set, and SCRSSN contains the value entered by the user. Once the data are subset, FETCH reads the observation.

```
whererc=where(dsid,'ssn='||scrssn);
rc=fetch(dsid);
```

To determine if the fetched observation is locked, you can either check for a return code of −630054 or use the SYSRC macro as in the following code fragment:

```
if (rc=%sysrc(_swnoupd)) then do;
```

The system variable SYSMSG also reports the following warning:

```
WARNING: SASBBL (Server Connection 9) is using this observation.
```

At this point you know another user may be updating the observation with values that are different than what you read in, and any update you may attempt to the observation currently locked will be unsuccessful.

Be careful when executing the FETCH function when a WHERE clause has been cleared or the REWIND function has been executed. Both of these situations move the pointer to the first observation in the data set. A subsequent FETCH at that point would lock the first observation in the data set.

The FETCHOBS Function

The FETCHOBS function can also be used in conjunction with the WHERE function to take advantage of the more efficient method of subsetting data. However, if you attempt to fetch observation 1 and a WHERE clause is active, SAS software fetches the first observation of the WHERE subset rather than the first observation in the data set. For this reason, the FETCH function is more often used.

The FETCHOBS function reads the specified observation and attempts to gain a lock on that observation. As with the FETCH function, you can either check for a return code of −630054 or use the SYSRC macro to determine if the observation is currently in use and to alert you that values you read may change.

The LOCATEC and LOCATEN Functions

The LOCATEC and LOCATEN functions are useful for performing a binary search on a small data set that is sorted by a unique variable. Otherwise, WHERE clause processing is more efficient in a multi-user environment. The LOCATEC or LOCATEN function locks the observation when the value is found if possible but does not indicate if the observation could not be locked. If this information is necessary, you may want to use the WHERE and FETCH functions instead.

THE DATALISTC and DATALISTN Functions

The DATALISTC and DATALISTN functions are useful for displaying values from a secondary data set in a selection list. Once you select a value from the displayed list, DATALISTC or DATALISTN attempts to gain a lock on that observation. As with LOCATEC and LOCATEN, DATALISTC and DATALISTN do not indicate if they were successful in gaining the lock. Also, even if you have CALL SET specified in your SCL program, SAS software only reads the value for the variable specified on the DATALISTC or DATALISTN function and does not map the values for other variables in the data set.

EXAMPLE OF A MULTI-USER FSEDIT SCL APPLICATION

The following is a multi-user SCL application using the FSEDIT procedure that assigns a unique id number to all new employees added to the database. The unique employee id number is stored in a secondary data set called NEWID that is defined to a server. As each data entry person adds a new employee, the SCL program reads the employee id number last issued from the secondary data set, increments the number by 1, updates the secondary data set with the new id number, and displays the id number on the new observation. In this example, you see an observation for each employee in the company, as shown in Display 2.

```
+FSEDIT SHARE.EMPLOYEE-------------------------------------------------Obs 1+
| Command ===>                                                            |
|                                                                         |
|                                                                         |
|  Employee ID Number:          1525                                      |
|                                                                         |
|  Full Name          :  SARAH M. ROBERTS                                 |
|                                                                         |
|  Title              :  SYSTEMS ANALYST              Extension:  3310    |
|                                                                         |
|  Manager            :  GINNY MCALLISTER             Dept code:  DPD     |
|                                                                         |
|  Home Address       :  250 WALKER DRIVE                                 |
|                        CARY, N.C. 27513                                 |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

Display 2

To add a new employee, issue the ADD command and press ENTER. Since CONTROL ALWAYS is specified, statements in the MAIN section of the SCL program execute to determine the unique employee id number. If the id number is locked so the program cannot retrieve a unique number, you remain on the current observation in the FSEDIT window, and a message displays telling you to try again, as shown in Display 3.

```
+FSEDIT SHARE.EMPLOYEE-------------------------------------------------Obs 1+
| Command ===>                                                            |
| ID numbers were locked. Please try again.                               |
|                                                                         |
|  Employee ID Number:          1525                                      |
|                                                                         |
|  Full Name          :  SARAH M. ROBERTS                                 |
|                                                                         |
|  Title              :  SYSTEMS ANALYST              Extension:  3310    |
|                                                                         |
|  Manager            :  GINNY MCALLISTER             Dept code:  DPD     |
|                                                                         |
|  Home Address       :  250 WALKER DRIVE                                 |
|                        CARY, N.C. 27513                                 |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

Display 3

If the SCL program is able to retrieve a unique employee id number, you are placed on a new observation where you can enter information regarding the new employee, as shown in Display 4.

```
+FSEDIT SHARE.EMPLOYEE-------------------------------------------------Obs 2+
| Command ===>                                                            |
|                                                                         |
|                                                                         |
|  Employee ID Number:          1526                                      |
|                                                                         |
|  Full Name          :  _____________________                            |
|                                                                         |
|  Title              :  _____________________    Extension:  ____        |
|                                                                         |
|  Manager            :  _____________________    Dept code:  ____        |
|                                                                         |
|  Home Address       :  _____________________                            |
|                        _____________________                            |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
|                                                                         |
+-------------------------------------------------------------------------+
```

Display 4

TRACING PROGRAM EXECUTION

The SCL program for this example follows.

```
  FSEINIT:
❶ dsid=open('share.newid','u');
  vid=varnum(dsid,'idnum');
❷ control always;
  return;
  INIT:
❼ if obsinfo('new') and idnum=. then idnum=symgetn('macid');
  return;
  MAIN:
❸ if word(1,'u') = 'ADD' or word(1,'u') = 'DUP' then
   do;
    count=0;
    link idloop;
   end;
  return;
❹ IDLOOP:
  count + 1;
  rc=fetchobs(dsid,1);
  if rc > 0 then
   do;
    call nextcmd();
    _msg_='OBS could not be added. Please try again.';
    return;
```

```
    end;
❺ if rc = -630054 then
    do;
     if count < 500 then goto idloop;
     call nextcmd();
     _msg_='ID numbers were locked. Please try again.';
     return;
    end;
❻ id=getvarn(dsid,vid);
   id = id +1;
   call putvarn(dsid,vid,id);
   rc=update(dsid);
   rc=unlock(dsid);
   call symputn('macid',id);
   return;
 TERM:
   return;
 FSETERM:
   if (dsid > 0) then rc=close(dsid);
   return;
```

The following list corresponds to the numbered sections of code.

1. Opening the NEWID Data Set - Open the NEWID data set defined to a server in update mode. Place the OPEN statement in the FSEINIT section of the SCL program so the OPEN will only be performed once when you invoke the FSEDIT session.

2. Executing MAIN - Specify CONTROL ALWAYS to allow statements in the MAIN section to execute before commands are processed.

3. Capturing Commands - When the ADD or DUP commands are issued, set your counter to 0 and link to the IDLOOP labeled section to attempt to assign a unique employee id number to variable IDNUM.

4. Fetching the Observation - Increment the counter by 1, and fetch the observation from the data set NEWID. There is only one observation in this data set since its only purpose is to hold the value of the last employee id number issued.

5. Checking the Return Code - A return code of −630054 from the FETCHOBS function indicates the observation is currently locked by another user. Continue to loop through the statements in IDLOOP until either you are able to successfully fetch the observation or the counter reaches 500. If the counter reaches 500 and the observation is still locked, clear the ADD command and display a message that the id number is locked.

6. Incrementing the ID Number - If FETCHOBS is successful, read in the value of the employee id last issued, increment the number by 1, and update the data set NEWID with the new value. Then unlock the observation so it will be accessible for the next employee being added, and place the new employee id number in macro variable MACID.

7. Assigning the ID Number - As each new observation displays, retrieve the unique employee id number from macro variable MACID.

CONCLUSION

This paper discusses locking considerations you should be aware of when writing FSEDIT SCL applications in a multi-user environment. SAS/FSP and SAS/SHARE software can now work together to enhance your FSEDIT applications and allow multiple people to edit the same data set at once.

ACKNOWLEDGMENTS

The author wishes to thank Yvonne Selby, Annette Harris, Mike Carney, Bill Brideson, Jennifer Boyle, and Kevin Hobbs for technical review.

REFERENCE

SAS Institute Inc.(1991), *SAS/SHARE Software: Usage and Reference, Version 6, First Edition*, Cary, NC: SAS Institute Inc.