Paper 195-25

Multiple Users to Update the Same Data Set in the Windows Environment

Quan Ren , Kendle International Inc. Cincinnati, OH

ABSTRACT

Windows operating system doesn't allow multiple users to update the same SAS® data set in the same time, however in the real life it is very often that several users need the right to update the same SAS data set in the same time. For instance, several people enter data into the same SAS data set in the same time; everyone needs the right to update the same data set. This paper is mainly to present an idea to solve this problem and make Windows to allow multiple users to access even update the same SAS data set.

INTRODUCTION

There are two levels of access to SAS data sets: read only access and read/write access. With local data libraries (Figure 1), there won't be any access conflict. With data libraries on network, if all access are read only then there won't be any problem (Figure 2), otherwise access conflicts could happen when several users try to access the same SAS data set in the same time. For example, user A has obtained the access to a SAS data set and is updating the data set, if in the same time user B sends a request to access the same data set, the request will be denied and an error message will be generated (Figure 3):

"ERROR: Member or library … unavailable for use."

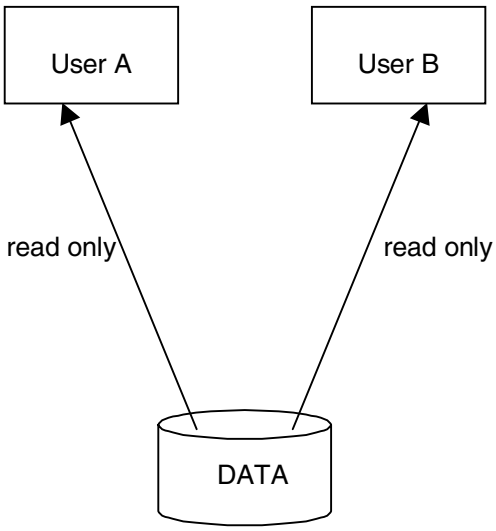


Figure 2. Read only access to network data

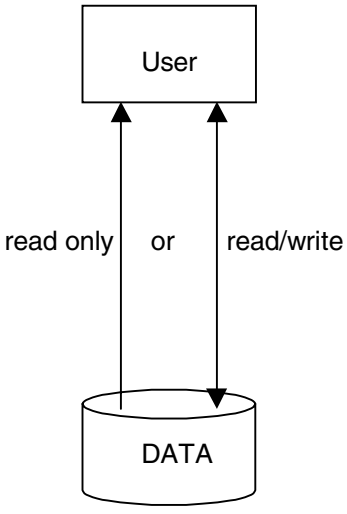


Figure 1. Access to local data

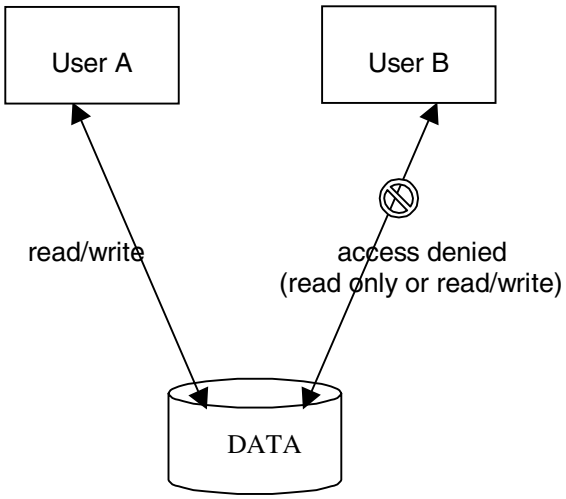


Figure 3. Read/write access to network data, access conflict could happen

Example 1.

Suppose User A is updating data set DAT1 on the network by submitting:

```
PROC SORT DATA=LIB.DAT1 OUT=LIB.DAT1 ;
    BY INDEX ;
RUN;
```

In the same time, if User B tries to update the same data set DAT1 by submitting the following:

```
PROC SORT DATA=LIB.DAT1 OUT=LIB.DAT1 ;
    BY   KEY ;
RUN;
```

DAT1 won't be updated by User B and the following error message will be generated:

"ERROR: Member or library … unavailable for use."

There are several ways to work around:

1. With SAS/Share® Software
2. Without SAS/Share Software

MULTIPLE ACCESS/UPDATE WITH SAS/SHARE SOFTWARE

With SAS/Share Software, a central SAS server manages all the data access requests from different users (Figure 4). When a user sends an access request to SAS server, the server will check all the current access to decide whether to grant access to the new request or not. If the new request won't conflict with any current access, access will be granted, otherwise the request will be denied and a warning message will be sent to the user.

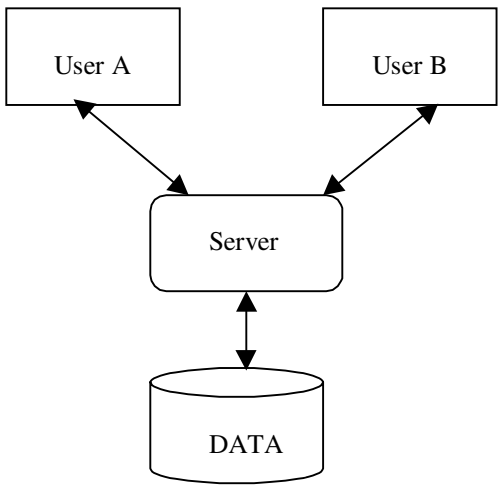


Figure 4. Access to network data with SAS/Share

Example 2.

Suppose that User A is updating data set DAT1 on the network by submitting the following code using SAS/Share:

```
%LIBREF(LIB) ;
PROC FSVIEW DATA=LIB.DAT1 ;
RUN;
```

here macro %LIBREF is used to make the network library available for use. Suppose that User A is editing observation 6. In the same time, if User B tries to update the same data set DAT1 by submitting the following:

```
%LIBREF(LIB) ;
DATA LIB.DAT1 ;
    SET LIB.DAT1 ;
    NEWVAR=_N_ ;
RUN;
```

Since observation 6 is locked by User A for editing, when User B reads observation 6, updating will stop and warning message will be generated:

WARNING: User A (server connection 1) is using this observation

If User B still wants to update the data set, he/she has to wait until User A finish his/her job and then resubmit the code. With the cost of SAS/Share, in this case the solution doesn't seem satisfactory.

MULTIPLE ACCESS/UPDATE WITHOUT SAS/SHARE SOFTWARE

Multiple access conflicts can be resolved without SAS/Share, in some cases the solution is even better. The following examples achieved the multiple access without SAS/Share.

Example 3.

In Example 1, if User B tries to update data set DAT1 by submitting the following:

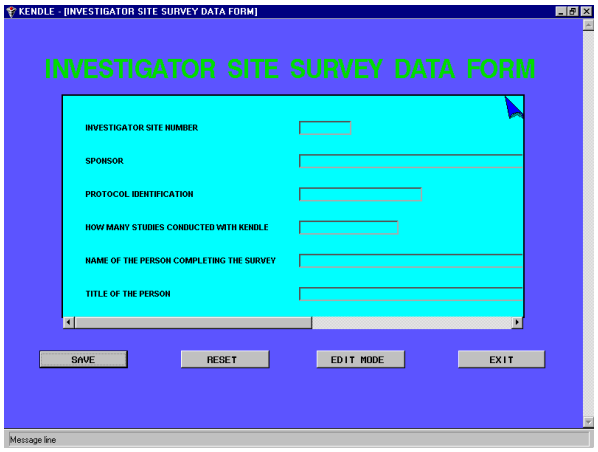
```
%MACRO ACCESS ;
    %LET   ERR= 99 ;
    %DO %WHILE(&ERR>4) ;

        PROC SORT DATA=LIB.DAT1 OUT=LIB.DAT1;
           BY   KEY ;
        RUN;

        %LET   ERR=&SYSERR ;
   %END;
%MEND;

%ACCESS ;
```

Using this piece of code to update DAT1, there won't be any errors. SAS System will automatically check if User A has finished updating and it will automatically update the data set for User B as soon as User A finish updating.

Example 4.

With SAS/AF® application, if there is a multiple access conflict, the error or warning message will hide behind the application screen; it's hard to know whether the access is successful or not.

Therefore, a guarantee for multiple access is indispensable. The following is an example to achieve multiple access with SAS/AF application. In this example, a simple data entry screen has been used to enter data into SAS data set QUESTFX. On this screen, there are a data form object QUESTINV and click buttons **SAVE, RESET, EDIT MODE** and **EXIT**.



The following is the SCL code to support this screen.

SCL code 1 :

```
INIT:

 CALL  NOTIFY('.' , '_GET_WIDGET_','QUESTINV' , FMID);

   SUBMIT CONTINUE;
     LIBNAME LIB 'C:\DIR' ;
   ENDSUBMIT ;

 CALL SEND(FMID,'_SET_DATASET_','LIB.QUESTFX',
                                              'EDIT','RECORD');

 CALL SEND (FMID,'_REFILL_USING_ATTRIBUTES_' , 'N' );
 CALL  SEND(FMID, '_ADD_ROW_' ) ;
 CALL EXECCMDI('ZOOM ON');

 MAXCOL=WINFO('MAXCOL');
 MAXROW=WINFO('MAXROW');
 CALL NOTIFY('.','_SET_WINDOW_SIZE_',1,1,MAXROW,MAXCOL);
 _MSG_='Message line';

RETURN;


SAVE:

 CALL  SEND(FMID, '_COMMIT_NEW_ROW_' ) ;
 CALL  SEND(FMID, '_REREAD_' ) ;
 CALL  SEND(FMID, '_ADD_ROW_' ) ;
 _MSG_ ='INFORMATION HAS BEEN SAVED INTO DATABASE';

RETURN;


RESET:

   _MSG_=' ';
    CALL  SEND(FMID, '_REREAD_' ) ;
    CALL  SEND(FMID, '_ADD_ROW_' ) ;
   _MSG_ ='SCREEN HAS BEEN CLEANED';

RETURN;


MAIN:

RETURN;


TERM:

   SUBMIT CONTINUE;
        LIBNAME LIB CLEAR ;
   ENDSUBMIT ;

RETURN;
```

With code 1, data set QUESTFX is used to start the data entry screen. Data set QUESTFX is locked as soon as a user logs in the application for data entry, if the data set QUESTFX is on a network, then no any other users can log in the application for data entry since data set QUESTFX is locked by the first user. In this case the application doesn't support multiple access. Therefore, the SCL code needs to be revised to fully support multiple access/update.

SCL code 2 :

```
INIT:

 CALL  NOTIFY('.' , '_GET_WIDGET_','QUESTINV' , FMID);

   SUBMIT CONTINUE;
    LIBNAME LIB 'C:\DIR' ;
    PROC DATASETS  LIBRARY=WORK NOWARN NOLIST ;
      DELETE TMP TMP2 TMP3/MEMTYPE=DATA ;
    RUN;

    DATA TMP ;
      SET LIB.QUESTFX(FIRSTOBS=1 OBS=0 ) ;
    RUN;

    DATA TMP2 ;
      SET TMP ;
    RUN;

    DATA TMP3 ;
      SET TMP ;
    RUN;
   ENDSUBMIT ;

 CALL SEND (FMID,'_SET_DATASET_','TMP','EDIT','RECORD');
 CALL SEND (FMID,'_REFILL_USING_ATTRIBUTES_' , 'N' );
 CALL  SEND(FMID, '_ADD_ROW_' ) ;
 CALL EXECCMDI('ZOOM ON');
 MAXCOL=WINFO('MAXCOL');
 MAXROW=WINFO('MAXROW');
  CALL NOTIFY('.','_SET_WINDOW_SIZE_',1,1,MAXROW,MAXCOL);
  _MSG_='Message line';

RETURN;


SAVE:

   CALL  SEND(FMID, '_COMMIT_NEW_ROW_' ) ;

   SUBMIT CONTINUE;
    DATA LIB.QUESTFX ;
      SET LIB.QUESTFX TMP(FIRSTOBS=1 OBS=1 ) ;
    RUN;
   ENDSUBMIT ;

   CALL SEND(FMID,'_SET_DATASET_','TMP3','EDIT','RECORD');

   SUBMIT CONTINUE;
    DATA TMP ;
      SET TMP2 ;
    RUN;
   ENDSUBMIT ;

   CALL SEND(FMID,'_SET_DATASET_','TMP','EDIT','RECORD');
   CALL  SEND(FMID, '_ADD_ROW_' ) ;
   _MSG_ ='INFORMATION HAS BEEN SAVED INTO DATABASE';

RETURN;


RESET:

    CALL  SEND(FMID, '_REREAD_' ) ;
    CALL  SEND(FMID, '_ADD_ROW_' ) ;
   _MSG_ ='SCREEN HAS BEEN CLEANED';

RETURN;


MAIN:

RETURN;


TERM:

   SUBMIT CONTINUE;
    PROC DATASETS  LIBRARY=WORK NOWARN NOLIST ;
      DELETE TMP TMP2 TMP3/MEMTYPE=DATA ;
    RUN;
   ENDSUBMIT ;

RETURN;
```

With code 2, the application makes an independent copy of the structure of data set QUESTFX first, then the structure copy is used to start the data entry screen. If there are several users, everyone can log in the application for data entry since everyone uses an independent data structure copy. In this way, multiple access to the application is supported. However, data access conflict still could happen when several users enter data in the same time. Multiple access/update to data set QUESTFX is not supported. For example, if User A entered some information into the data entry fields on the screen and clicked the **SAVE** button to save the information into data set QUESTFX; In the same time, if User B clicks the **SAVE** button to save the information that he/she just entered, the request from User B will be denied and the information will be lost since data set QUESTFX is locked by User A for saving. No warning or error message will show up on the screen and User B won't even know the access has failed. To solve this problem, the SCL code part **SAVE : … … RETURN** in code 2 needs to be revised :

```
SAVE:

 CALL  SEND(FMID, '_COMMIT_NEW_ROW_' ) ;

  ADD:

   SUBMIT CONTINUE;
    DATA LIB.QUESTFX ;
      SET LIB.QUESTFX TMP(FIRSTOBS=1 OBS=1 ) ;
    RUN;
   ENDSUBMIT ;

   ERR=SYMGET("SYSERR") ;
   IF ERR>4 THEN GOTO ADD ;

   ERR=0 ;

   CALL SEND(FMID,'_SET_DATASET_','TMP3','EDIT','RECORD');

   SUBMIT CONTINUE;
    DATA TMP ;
      SET TMP2 ;
    RUN;
   ENDSUBMIT ;

   CALL SEND(FMID,'_SET_DATASET_','TMP','EDIT','RECORD');
   CALL SEND(FMID, '_ADD_ROW_' ) ;

   _MSG_ ='INFORMATION HAS BEEN SAVED INTO DATABASE';

RETURN;
```

With the revised code 2, there won't be any access conflict, multiple access/update is fully supported.

TRADEMARKS

REFERENCES

SAS Institute Inc. (1990), *SAS Guide to Macro Processing: Version 6, Second Edition*, Cary, NC: SAS Institute Inc.

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

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact the author at:

Quan Ren
Kendle International, Inc.
1200 Carew Tower
5th & Vine Streets
Cincinnati, OH 45202 USA
(513) 763 -1318
E-mail: qruiyu@aol.com *or* ren.quan@kendle.com