# DATA SHARING USING SAS/DB2™ SOFTWARE

ANDREW S. SCHLEGEL
BARBARA E. LEE

## INTRODUCTION

Since our company does not have SAS SHARE we have had to deal with the problem of data sharing by another means. In the past we were able to set up data sets for each of our four divisions and alleviate some of the problem. As more systems were created and the complexity of the systems grew, this alternative was no longer effective. A new solution had to be created to allow for data sharing within the SAS system.

DB2 was selected as the alternative when the SAS DB2 product was brought in. The initial concept we had for using this product was a little naive. We intended to create our data files in DB2 and use SAS DB2 to access the data. The first attempt at this was to define our data files as normalized as possible. We then tried to use FSEDIT on the tables to update the data. Our first problem was how to update dependant data or joined tables. We found out that only one table could be updated at a time, and no views. Well, we were able to work around this problem as best we could. We then tried to allow multiple updates to the same table. Look Out!

Here is where the big problem came in. DB2 locks out updates against a table when it is already being updated. This would have been OK if it would then free the table as soon as it was done. It did not, instead you could not even access the table and the FSEDIT procedures would abend. Other problems occured with data integrity.

What do we do now? A different approach had to be created to allow data sharing and it had to be done using SAS and DB2 since there were no alternatives for us to use. The following paper describes how we accomplished this task to make a data sharing system using the benefits of both SAS and DB2.

## BACKGROUND

We will use our Corporate Property Records System as a sample of how we made this work. This is the second system that we installed using this processing. This system tracks our distribution facilities by grid number which identifies their location. This includes the poles, wires, street lights, etc. The system needed to be updated by each of our 4 divisions by numerous personnel. It also had to be available to our 16 districts and various departments with the most up to date information available. The number of records in the main table would be around 400,000 records. We determined that our normal SAS data sharing methods would not handle this type of system. So we proceeded to create DB2 tables to be accessed by SAS.

## THE DB2 TABLES AND VIEWS

The following is a list of the DB2 tables that we are using in our example. The actual system uses 12 tables to maintain all of the different attributes of the CPR system.

- Poles - This is a table with both history and current information on the poles at each location.

- Grid - This table has the grid numbers and associated information about the location in the field.

## THE SAS DB2 PROCESSING

In processing this system there are a few transactions that take place. They all follow the same processing. They are:

- Conversion - Enter all information at a location.

- Updates - Updates the information stored in DB2 tables.

- Browse - These transactions display the selected data.

You can select any of these records by entering a grid number.

## CONVERSION PROCESS

The first processing we will look at is the conversion screen processing. In this process we had to set up a system to allow from 4-12 keypunchers to input data into the CPR system. We did this by creating a SAS/AF application with 4 screens. The screen is displayed and each of the fields have associated macro variables defined.

As the user enters data into the screens and then presses ENTER or the END PF key, the AF screen processes the data. We used the initial macro feature of the AF program screen to do the majority of our error processing. This will not allow the user to leave the screen if any errors occur. The only processing we could not accomplish in the the initial macro procedure was DB2 validations. We did this in the program area of SAS/AF.

Note: One new feature we used in setting up the FSEDIT screen was the Invalue option of Proc Format. We did this to verify the CPR number, dates, and also to display the description of the CPR number. An example of the pole code check is shown below:

```
PROC FORMAT;
   INVALUE $CPRP '0050357' = '0050357'
                 '0050400' = '0050400'
                 '0050455' = '0050455'
                     ' '   = ' '
                   other   = 'ERR';
```

By setting up the invalue with this format it verifies that the value on the right is valid and changes it to itself. If it cannot find the value then it is an error. If the field is not a required field make sure that a blank = blank is in the list.

The keypunchers enter a grid number for a location. The system first tries to retrieve the data from the grid table using PROC DB2EXT. If it finds a record it creates macro variables for the data it retrieved and displays the next AF screen. If it doesn't find a record an error message is displayed.

When the keypunchers are finished inputting the data they press ENTER or use the END PF key. At this time the invalue formats are processed and the initial macro processes. The initial macro verifies that all fields were entered correctly. It also checks cross field validations that cannot be done in FSEDIT. This helps to insure that all of the data is correct before continuing. If the user presses the END PF key and there are no errors the system will then proceed to the program part of the AF screen. Here the macro variables are mapped to SAS variables and data sets are created for each of the tables to be updated.

A SAS macro is then called to update the DB2 tables. It checks to see if data is entered for a table and then uses PROC DB2UTIL to insert the records into the table. It then checks the SAS system macro &SYSERR to insure that the DB2 processing completed successfully. If an error occured such as duplicate insert, the processing stops and an error message is displayed on the screen explaining the problem.

## UPDATES

The CPR system has numerous update functions. You can update all information for a specific location or you can update a specific type of data for a location. Both processes work basically the same. The first process that happens in both updates is the user enters a grid number for updating. Again the system checks to see if it is a valid grid number as it did above. If it is a valid grid number the system would then proceed to extract the appropriate information from the DB2 tables using DB2EXT. This happens fairly quickly when going against a specific set of data because it only has to run one DB2EXT. In the summary update where the user updates all of the information for a location, the system has to retrieve data from several DB2 tables.

When the data is retrieved the system creates macro variables for the data it selected and a SAS/AF screen is displayed for updating the data.

The process proceeds as above except when it creates SAS data files it creates 2 for each table. One is for updates and the second is for inserts. A macro is again called to perform the updates and inserts using DB2UTIL as required.

## BROWSE TRANSACTIONS

All of the browse transactions work in the same manner by first extracting the data using the DB2EXT procedure. It then creates macro variables for all of the fields and displays a SAS/AF screen.

## THINGS WE HAVE LEARNED

In trying to use DB2 with SAS in a shared environment we have learned a few key things which we have listed for your use:

Do not use FSEDIT directly against a DB2 table. Aside from the lock out problem we ran into, integrity of the tables also seemed to be vulnerable. There were occasions when data was dropped from tables and other cases when duplicate rows were

added. Because of these problems we have made a policy of not using FSEDIT against a DB2 table.

Do not use FSBROWSE against a large view or table. The problem we found using this process was that when viewing a large table it takes quite a bit of time to move around within the table. This is caused by having to read through the entire data base from beginning to end if you want to view records in front of your current location. A simple backward command may take a few minutes if you are browsing a large table.

Be careful when inserting data into multiple tables at the same time if you put constraints on either of them. We ran into this problem with a prior project when in the conversion process we constrained the serial number to be unique. Make sure that you check the SAS macro variable &SYSERR before entering data into a second table if it is a dependent of data in the first.

Both DB2LOAD and DB2EXT take excessive CPU time. When trying to load a large number of records into a DB2 table using DB2LOAD it required considerable more CPU time than if we used the DB2 load utility. The same thing happened when trying to extract a DB2 view from multiple tables.

Watch out for null values. Version 5.18 seems to fix most of the problems we had with this but just be careful. Also a problem with release 5.18 causes your DB2 map data sets to not work with the prior release and vice versa.

Limit the number of tables to 2. In developing these systems we found problems with the speed of some of the transactions. The way SAS process DB2 calls creates some problems. Every time you issue a DB2 procedure SAS reconnects itself to DB2. This can take quite a bit of time if you issue multiple calls. A suggestion was given to SAS to allow a parameter to be passed to SAS at invocation time to tell it to connect to DB2 and stay connected until you leave SAS. This would allow for much faster processing time against multiple tables or in transaction processing.

## BENEFITS WE HAVE GAINED

We did see major benefits in using this procedure and they are listed below:

1. **We did create a shared system.** By using this procedure we were able to create a shared system that allowed updates from many locations into the same data base. While we had an average of 8 keypunchers entering daily data, we also had the four divisions maintaining the data. There were numerous other departments from the 16 districts viewing the data along with running SQL reports concurrently.

2. **Ease of use.** We were able to put up a system that was very friendly for our users. The AF front-end was much easier for them to use than the normal IMS/ADF™ screen processing. The only problem was getting used to pressing the submit key to process a transaction rather than the Enter key.

3. **Ease of Creating.** When we finally figured out how to run SAS against a DB2 table the programming of the system became very simple and we were able to create the system in a much shorter time span than the normal IMS/ADF processing used in the past. New features can be added to the system with minimal effort. The system can be created in a prototype fashion, but you have a production system when completed.

4. **Use of Indexing.** The ability to use DB2's indexing features within SAS made the system very efficient in retrieving information. Even though the speed is not as fast as using ADF it does perform satisfactorily.

5. **Using SQL against the tables.** One of the big benefits of the system is the ability to use DB2 SQL reporting. Even though reporting in SAS is not a difficult task, reporting with SQL is even easier and the users were writing reports within a very short time.

6. **Transaction Logging.** The DB2 logging facility helps to maintain the data integrity of the system. If the system fails, DB2 will rollback and then process the transaction log to keep the tables accurate.

7. **Online Validation.** Using this process with SAS/AF we can produce online field validation with cross field checks. With indexing this process can be done efficiently and helps to keep our data as valid as possible.

## CONCLUSION

Even though we have had to work around problems to come up with this method of creating a data sharing system, I think the

benefits we have gained from this system have outweighed the headaches. We are now able to use the relational power of DB2 with the ease of use of SAS to create a system that makes it easy for our users to maintain the data. They are also able to use the data by writing DB2 SQL reports to answer the "what if" type questions that arise. This is our design approach for the future or until VERSION 6.0 for MVS with indexing and relational data processing is installed at our facility.

FOR MORE INFORMATION
Andy Schlegel or Barbara Lee
Metropolitan Edison Copany
P.O. Box 16001
Reading, PA 19640

SAS, SAS/AF are registered trademarks of SAS Institute Inc., Cary, NC, USA.

IMS/ADF, DB2 are registered trademarks of International Business Machines Corporation.