ABSTRACT

Although the internal storage structures of SYSTEM 2000® software and SAS® data files differ, they are quite compatible in the SAS environment. Interfaces are provided to ensure efficient and accurate transitions from one to the other. This paper addresses the reasons for choosing one form over the other and gives methods for migrating between a SAS data set and a SYSTEM 2000 database.

INTRODUCTION

SYSTEM 2000 software is a hierarchical data management system that is available on mainframe computer systems. When SYSTEM 2000 software interfaces with the SAS System, it only operates on IBM® mainframes in MVS and CMS environments. With SYSTEM 2000 software you define the types of data, group the data, and determine the relationships among the groups. The structure that results resembles a family tree.

SYSTEM 2000 Structure

DEFINITION

EMPLOYEE

POSITION

DATA

JOHN JONES

Manager Sr Programmer Programmer

The SAS System maintains its basic data information in data files, which can be thought of as tabular in nature. That is, the data values are stored in a rectangular form and the descriptor portion keeps information about the values much like the row across the top of a table.

SAS Structure

EMPNAME POSITION +Descriptor

JOHN JONES MANAGER

JOHN JONES PROGRAMMER

JOHN JONES SR PROGRAMMER +Data

REQUIREMENTS THAT INDICATE SYSTEM 2000 USAGE

SYSTEM 2000 software provides a variety of capabilities that augment the SAS System. The benefits of SYSTEM 2000 software apply to Releases 5.18 through 6.06 of the SAS System.

Reduction of Storage

Many applications fall naturally into hierarchical structures that allow one-to-many relationships as well as one-to-one relationship provided by relational systems. Like a family tree, as you move down the tree, none, one, or many children can occur. In SYSTEM 2000 software, storage space is only used if a record occurs.

SAS data files can be considered to be relational in that they are stored in rows and columns. With this type of structure the data must be filled in for each column, which can produce redundant data when the same information occurs for a column. This redundancy is inherent in relational systems and can translate directly into wasted space. You can recognize this situation if you sort your data and print them. Screen 1 illustrates redundancy due to the tabular nature.
Example 1: Redundancy Due to Tabular Form

/* Sort the data */
PROC SORT DATA=EMPPOS;
   BY LASTNAME, FIRSTNAME, POSITION, MANAGER;
RUN;
/* Print data & examine output */
PROC PRINT DATA=EMPPOS;
RUN;

The SAS System produces output such as that shown in Screen 1.

<table>
<thead>
<tr>
<th>OBS</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AHEER</td>
<td>DAVID</td>
<td>PROGRAMMER</td>
</tr>
<tr>
<td>2</td>
<td>AHEER</td>
<td>DAVID</td>
<td>SR SALES REPRE</td>
</tr>
<tr>
<td>15</td>
<td>GARRETT</td>
<td>OLAN M.</td>
<td>SR SALES REPRE</td>
</tr>
<tr>
<td>16</td>
<td>GARRETT</td>
<td>OLAN M.</td>
<td>MANAGER OF SALES</td>
</tr>
<tr>
<td>17</td>
<td>GARRETT</td>
<td>OLAN M.</td>
<td>VICE-PRESIDENT</td>
</tr>
</tbody>
</table>

Screen 1 Output with Redundancy

Designing to Alleviate Redundancy

You can design SAS data files and try to reduce repetitive information by creating different variables for separate occurrences of the same field. This design might be appropriate for a specific application, but it might not be effective for performing statistical analysis.

An example of a cumbersome design might be to represent the salary information with twelve different variables as follows

EMPINFO PAY1 PAY2 ... PAY12.

This definition works well if you want statistics for one month's pay, such as totalling PAY2 to produce February expense reports. It is not the best design for providing year-to-date totals for each employee as shown here

TOTALPAY = PAY1 + PAY2 + PAY3 + ... PAY12;

SYSTEM 2000 software alleviates redundancy by its structure. It keeps information about the relationships between groups, or records.

To improve on the previous design, you can use SYSTEM 2000 software to define two records, EMPLOYEE and PAYROLL, with the payroll information occurring many times for each employee. This design saves on storage space because the information is only required once per employee and payroll information does not take up space if it does not occur for an employee.

This design allows you to produce both January expense reports and year-to-date reports, as follows

PRINT SUM PAY WHERE MONTH = JAN;
PRINT SUM PAY BY EMPLOYEE;

If you do not want to use space to store the month, you can represent the month information by logical placement of the payroll information under the employee. The first payroll group would be for an employee's January pay information, the second for February, and so on.

EMPLOYEE Jones
PAY Jan Feb Mar ..... Dec

As above, the yearly payroll total for each employee can be obtained with:

PRINT SUM PAY BY EMPLOYEE;

To calculate amount paid to all employees for the month of January, use the logical relative month number as follows

PRINT SUM PAY WHERE PAY EXISTS AT 1;
Saving Storage with Extended Fields

In many relational systems each field is fixed in length. You specify the maximum acceptable size for each character field, and that amount of space is reserved for that field each time a row, or observation, occurs. Values that are shorter than the specified length result in wasted space, and values that exceed the length are truncated.

A good example of wasted space would be the LASTNAME field of an EMPLOYEE data set. Approximately 80% of surnames in the United States have fewer than ten characters. In many relational systems, you would reserve space for the longest acceptable value, say fifteen characters. This means that five characters will be wasted for 80% of the data and some values will not be fully stored.

SYSTEM 2000 software allows you to reserve the optimal amount of 10 characters but accepts lengths up to 250 characters. The extension is automatic and transparent as long as the field has at least four characters defined.

Release 6.06 of the SAS System provides a similar capability with a COMPRESS data set option. This option collapses blanks to produce varying length records.

Associating Two or More Data Sets Together

If two or more data files are usually associated together, they should probably be stored together, possibly in a SYSTEM 2000 database. You can save on processing time required to sort and merge the data files by maintaining the relationship in the structure.

Example 2: Wasted Processing Time

An example of possible wasted processing time would be the repetitive use of a sequence such as

```
PROC SORT DATA=EMPLOYEE;
  BY SSN;
RUN;
PROC SORT DATA=PAYROLL;
  BY SSN, PAYMONTH;
RUN;
```

Concurrent Access, Including Update

SYSTEM 2000 Multi-User™ software ensures data protection during concurrent updates and automatically guards a database against conflicting tasks. For most processes, SYSTEM 2000 software locks out at the record level. SYSTEM 2000 software also performs multiple holds, allows framing of commands, and performs user-specified rollback. It coordinates all activities of multiple users, including optional automatic rollback and recovery.

In the PROC statements you specify Multi-User in Release 5.18 with ACCESS=M, and in Version 6 with S2KMODE=M. If you do not specify the usage in the PROC statements, the screen or window indicates your selection choices, including Multi-User or single-user environments.

Additional Security

The SAS System provides security through views in Version 6 as well as through the operating system facilities. SYSTEM 2000 software provides additional security through a multi-level password system. This password system gives varying authorities assigned by the master password user for each record or item in the database.

The three types of password available are

- master password - unqualified access
- secondary password - retrieval, update, where-clause, or no authority assigned for each item in the database
- DBA password - administration capabilities without access to the data.
Optional Transaction Journals and Recovery

SYSTEM 2000 software provides optional transaction journals and rollback recovery from system crashes or other failures.

System Tuning Capabilities

With SYSTEM 2000 you get a comprehensive set of diagnostic and tuning capabilities. If you have special knowledge of your data, you can optimize processing and storage in a variety of ways. Some of the more relevant ways are:

- assigning blocksize on an individual basis to the various parts of the database, such as the data, index, relationship table, and so forth
- logical placement of the data based on some data value, such as a date
- reserving space for anticipated information
- adjusting run-time parameters.

Subsetting the Data

Both SYSTEM 2000 software and the SAS System support index structures to allow direct access to subsets of the data. There are trade-offs in the way the SAS System handles complex qualification statements versus the way SYSTEM 2000 software processes the equivalent requests. Examine the resource requirements using a prototype SAS data set and a SYSTEM 2000 database to determine which is better.

REQUIREMENTS THAT INDICATE SAS USAGE

The SAS System offers over 300 procedures for reporting, performing statistical analysis, producing graphics, and so forth. One of the best reasons to choose a SAS data file for storage is to execute the procedures in the most efficient manner. The SAS System performs most effectively using its own data structures.

Using Version 6 of the SAS System

The engine architecture of Version 6 enables you to store your data in SAS data files, SYSTEM 2000 databases, and other database structures. The SAS procedures can directly address the data without first extracting the data and creating SAS data sets.

To use SYSTEM 2000 software in Version 6 of the SAS System, you must define special files that describe the SYSTEM 2000 database and data to the SAS System. These files are called SAS/ACCESS® descriptor files. There are two types of descriptor files:

- access descriptor files - files with member type access
- view descriptor files - files with member type view.

Access descriptor information is kept in a SAS data library. You can only have one library for all of your access descriptors.

You create an access descriptor with the ACCESS procedure. In the Access window, you indicate that you want to create the access descriptor and you assign a three-part name with the libref, the new access descriptor name, and the type access (see Screen 2).

PROC ACCESS;
RUN;

Screen 2 Access Descriptor
The Access Descriptor Identification window appears. In this window you decide which database to use and supply a password.

Next the Access Descriptor Display window appears. In this window you decide what item information will and will not be displayed to users who create view descriptors from the access descriptor.

A view descriptor contains either complete or partial information about the SYSTEM 2000 database you want to use. This view descriptor is usually a subset of access descriptor information. You cannot have a view descriptor without a corresponding access descriptor.

You create a view descriptor using the ACCESS procedure. In the Access Window, you place your cursor on the selection field of the access descriptor, such as MYLIB.EMPLOYE. Then you type CV (create view) and press ENTER. Refer to Screen 3.

```
PROC ACCESS;
RUN;
```

You can optionally specify selection criteria using a SYSTEM 2000 where-clause and a SYSTEM 2000 ordering clause.

**Screen 3 Creating View Descriptor**

The View Descriptor Display window appears and you are shown everything that the access descriptor allows you to see. You then select or delete the items you want to use when you access this view.

**Using Release 5.18 of the SAS System**

Most SAS procedures in Release 5.18 can only access SAS data files. Any time you want to perform one of these SAS processes, you must first extract the relevant data into a SAS data set.

The SAS System provides macros and interfaces for using SYSTEM 2000 software in the SAS environment. Two procedures can be used for extracting SYSTEM 2000 data and creating SAS data files. One of these is the S2K procedure. The other, the FSEDIT procedure, uses the OUTPUT command to create SAS data sets. PROC S2K is for extraction only, but you can extract, browse, or update using PROC FSEDIT.

**TRANSFERRING DATA BETWEEN SAS DATA SETS AND SYSTEM 2000 DATABASES**

If you choose to transfer data between SYSTEM 2000 databases and SAS data sets, you invoke a CLIST or EXEC to execute the SAS System with SYSTEM 2000 software. As such, you have all the capabilities of SAS Display Manager System plus access to SAS and SYSTEM 2000 procedures.
Migrating from SAS Data Files to SYSTEM 2000 Databases

The following chart gives an overview of the methods available for creating a SYSTEM 2000 database from a SAS data set.

<table>
<thead>
<tr>
<th></th>
<th>Release 5.18</th>
<th>Version 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>MAKES2K</td>
<td>MAKES2K</td>
</tr>
<tr>
<td>syntax</td>
<td>macro</td>
<td>macro</td>
</tr>
<tr>
<td>define</td>
<td>PROC QUEST</td>
<td>PROC DBLOAD</td>
</tr>
<tr>
<td></td>
<td>PROC FSEDIT</td>
<td>PROC QUEST</td>
</tr>
<tr>
<td>load</td>
<td>PROC S2KLOAD</td>
<td>PROC DBLOAD</td>
</tr>
<tr>
<td></td>
<td>PROC FSEDIT</td>
<td>PROC FSEDIT</td>
</tr>
<tr>
<td></td>
<td>PROC QUEST</td>
<td>PROC REQUEST</td>
</tr>
<tr>
<td></td>
<td>PROC APPEND</td>
<td>PROC SQL</td>
</tr>
<tr>
<td></td>
<td>PROC FSVIEW</td>
<td>PROC FSVIEW</td>
</tr>
<tr>
<td></td>
<td>SAS/AF®</td>
<td>Software</td>
</tr>
</tbody>
</table>

Considerations for Design

There are three factors involved in any design choice — ease of use, storage space, and processing time. Your design can correspond to the SAS data file directly, or you can create multi-level structures with indexed items to take advantage of SYSTEM 2000 characteristics.

Factors that affect storage space are your overall design and extended field feature of SYSTEM 2000 software. A one-to-many design prevents redundancy of the higher level information and does not require space for descendant information if a record does not occur.

The extended field capability offers the ability to choose an optimal size for CHARACTER or TEXT fields. It accepts values that exceed that length, up to 250 characters. The SAS data set COMPRESS option provides similar functionality with varying length records.

Factors that affect processing time are indexing and number of levels in the design. Items, or fields, that are indexed are called key items in SYSTEM 2000 software. They provide quick access to the data records just as the index at the back of a book provides quick access to pages. You can choose to use the CREATE INDEX statement at any time. You can also use the REMOVE INDEX statement if you want to reuse space occupied by an unused index.

Most SYSTEM 2000 databases have fewer than four levels. Databases that have more levels may take more processing time. However, if you have infrequent data, you can actually save on processing time as well as storage space by placing that data at a lower level.

Another consideration in your design is that only one path of information - one branch of the definition - can be accessed at one time with the SAS procedures, except for the QUEST and SQL procedures. PROC SQL is available in Version 6 and provides for a join of two or more data files and/or data views.

In the example below you can either extract or update the EMPLOYEE/PAYROLL path or the EMPLOYEE/SCHOOL path in one execution of a procedure. You can also join the PAYROLL and SCHOOL information using PROC SQL in Version 6 of the SAS System.

EMPLOYEE
    PAYROLL
    SCHOOL

Ease of use depends on the individual. In general, simpler designs are easier to understand. Also, meaningful names are important. In the SAS System, the field names are limited to eight characters. SYSTEM 2000 software allows field names from 1 to 250 characters and accepts embedded blanks.

Letting the Interfaces Do the Work

The easiest way to create a database from a SAS data file is by using the MAKES2K macro. This macro differs in Release 5.18 and Version 6. In both versions you can simply invoke the macro, but the processing that occurs is different.

Executing MAKES2K with the defaults produces a one-level database that looks like the original data file. The names, types, and lengths are
the same as in the data set. The advantages of using SYSTEM 2000 software are the Multi-User coordination, additional security, and logging and recovery capabilities.

Using the MAKES2K Macro in Release 5.18 of the SAS System

In Release 5.18 of the SAS System, the MAKES2K macro creates a file containing syntax to

- allocate files for the new database (for TSO)
- assign a database password and name
- define the database using PROC QUEST
- copy the data values from the SAS data set by invoking the S2KLOAD procedure.

Steps for Using MAKES2K in Version 5.18

1. Log on to your system.
2. Invoke the SASS2K CLIST or EXEC that brings up display manager.
3. Work with only a subset of your data set for prototyping.
4. Sort your SAS data set if you plan to use multiple levels in your database.
5. Make the MAKES2K macro available for your session.
6. Call the MAKES2K macro. Specify the data set name, the database name, and the desired master password as follows

   `%makes2k (data = libref.data set, 
   dbn = new database name, 
   pw = new master password);`  

If you want to index some of the fields, issue

   `%makes2k (data = libref.data set, 
   dbns = new database name, 
   index = (item1, ...itemn));`  

MAKES2K creates an external file named DEFINE. You can save this file, edit it, and submit it.

7. Issue an INCLUDE command specifying the DEFINE file:

   `Command ==> include DEFINE`

The DEFINE file appears on your screen, and you can scroll through the statements that create this database. You can revise these statements if you want to change the data structure or definitions.

8. Press the SUBMIT key to submit the DEFINE file. After the database is created, a message appears in the SAS log window that tells how many observations were read and how many data records were inserted.

Using the MAKES2K Macro in Version 6 of the SAS System

In Version 6 of the SAS System, the MAKES2K macro creates a file containing syntax to

- allocate files for the new database (for TSO)
- assign a database password and name
- invoke PROC DBLOAD to create the database, an access descriptor, and a view descriptor, and copy the data values from the SAS data set.

Steps for Using MAKES2K in Version 6

1. Log on to your system.
2. Invoke the SASS2K CLIST or EXEC that brings up display manager.
3. Work with only a subset of your data set for prototyping.
4. Sort your SAS data set if you plan to use multiple levels in your database.

5. Make the MAKES2K macro available for your session.

6. Call the MAKES2K macro. Specify the data set name, the database name, and the desired master password as follows:

   \%
   makes2k (data = libref.data set,
              dbn = new database name,
              s2kpw = new master password);

   If you want to assign descriptor names and index some of the fields, issue:

   \%
   makes2k (data = libref.data set,
              dbn = new database name,
              s2kpw = new master password,
              viewdesc = new view descriptor,
              accdesc = new access descriptor,
              index = (item1, ..., itemn));

   MAKES2K creates an external file named DEFINE. You can save this file, edit it, and submit it.

7. Press the SUBMIT key. After the database is created, a message appears in the SAS LOG window that tells how many observations were read and how many data records were inserted.

Revising the Syntax in Either Version

If you want to take advantage of some of the features of SYSTEM 2000 software, you can modify the generated syntax before you submit it. You can revise the code with an editor or bring it into the display window and change it. Submit the revised statements either interactively or in batch mode.

If you want to change the code in display manager window, issue:

   Command==> include DEFINE

If you have chosen a different name by using the DEFINE= parameter in the MAKES2K macro, issue an INCLUDE command for that name.

Some good modifications might be to change component names, add indexes, change lengths, and provide level statements. A sample of revised code is:

   . { Allocations unchanged }
   .   PROC DBLOAD DBMS=S2K DATA=fordb;
       DBN=MYS2KDB;
       S2KPW=MYPW;
       S2KLOAD;
       LOAD;
       label
       s2klen lastname=6 firstname=5
       level paymonth=1 pay=1
       index lastname=y paymonth=y
       RUN;

   The generated code varies between the two releases of the SAS System. You should review the code and read the documentation for your release. The pertinent sections in Release 5.18 are PROC QUEST and PROC S2KLOAD. In Version 6, you should read the section for PROC DBLOAD.

Adding Indexes at a Later Time

If you choose the defaults, you can add indexes later using the QUEST procedure as follows:

   PROC QUEST ACCESS=S;
       USE, mypw;
       DBN mydbname;
       CONTROL;
       CREATE INDEX item, ..., item;
       EXIT;

SUMMARY

You have many choices when determining how your data is stored. If you use SYSTEM 2000 software, you can select from a number of procedures for creating your views or loading your data. The easiest way to get started is to use the MAKES2K macro and revise the code which it generates to suit your application.
REFERENCES

SAS Institute Inc. (1989),

IBM is a registered trademark of International Business Machines Corporation, Armonk, NY, USA.

SAS, SAS/ACCESS, SAS/AF and SYSTEM 2000 are registered trademarks of SAS Institute Inc., Cary, NC, USA.

Multi-User is a trademark of SAS Institute Inc., Cary, NC, USA.