A USER'S VIEW OF SAS/IMS-DL/I

Emilio A. Icaza, Louisiana State University

The Louisiana State University Computer Center is the nucleus of computing activity on the campus. A large IBM processor supports a variety of needs for the instructional, academic, research, and administrative user of the Baton Rouge campus. In addition, other campuses of the LSU System complement their computer needs with services provided by the Baton Rouge campus.

At last inventory there were about 95 different software products installed on the main computer and well over 2500 programs which service the administrative computing needs of the university. Among all the software, SAS is by far the most widely used product. In terms of executions it ranks second only to the Linkage Editor. In terms of volume, it is said to consume one month out of a year's supply of computer resources. The Information Management System (IMS) is the only large scale data base management system in use at LSU. Thus, the announcement of availability of SAS/IMS-DL/I brought a world of interest from all quarters of the LSU user community. This paper describes a Support Center's view of SAS/IMS and our experiences introducing the product to the LSU users.

Who Uses SAS?

Whenever a product like SAS/IMS-DL/I becomes available, someone in the organization has to identify the potential users of the product and then justify to the budgetary officers its acquisition. Invariably the question that gets asked is, "Who uses SAS?"

In the earlier years of SAS, its use was limited to research. The Department of Experimental Statistics acquired the basic product and continues to support SAS for the rest of the University. But as SAS Institute expands its product line, many more non-research applications of SAS are surfacing. A detailed list of users is beyond the scope of this paper. However, an understanding of the SAS user community needs and capabilities is fundamental for the success of a product such as SAS/IMS-DL/I in the university environment. SAS users at LSU can be classified into four groups:

1) Research Users. Hit harder by budgetary constraints, these users are ingeniously coming up with ways to stretch their computer dollars using SAS. In general, they are in the forefront of the university in terms of computer technology. A few have hired professional programmers, but most operate with graduate assistants. These users might not know about IMS but will probably use it if it is made available in a more palatable way.

2) Instructional Users. These users are interested in SAS specialty features rather than in its data management potential. They use SAS as a tool to teach statistics (PROC MATRIX), econometrics and business (SAS/ETS) and social science (PROC GLM, FACTOR, CORR). They need data base management teaching tools, but may lack the time or interest to digest IMS.

3) Administrative Users. These are the users that SAS/IMS was acquired for. They have the advantage that their data is already stored in IMS data bases. They want to use SAS to analyze the data, but they are the least technically prepared for it.

4) Programming Support Users. These include all analysts and programmers who support administrative system development and other computer center functions. Very few of these users have SAS programming knowledge and most are primarily one-language programmers. They can very easily learn to use SAS/IMS and would benefit greatly from it; however, they need encouragement.

The task for the Support Center group is to define a strategy to make a product such as SAS/IMS useful to a larger proportion of all or some of the user groups. But, in the end, both a knowledge of SAS and IMS is required.

What is IMS and Why Use IMS?

The latest IBM IMS/VS General Information Manual defines IMS as "...data base/data communications system capable of managing complex data bases and terminal networks". This definition contains components that need explanation.

First, a data base is a collection of data items usually stored on a direct access device. However, other storage devices such as tapes are also used. Data is organized in the data base according to the rules established by the data model. IMS uses the hierarchic data model. If we think of a data base as a collection of observations or records, the hierarchic organization is accomplished by grouping the data values in an observation into subsets called segments. To illustrate, consider a SAS data set of student information kept at a college (See Figure 1).

In the STUDENTS data set there are four types of information present: a) personal and cumulative information about the student; b) semester information on the student's attendance; and c) detail information on the course work taken and the student's performance.
Working with the students data set in SAS would be "a pain in the neck". In IMS however, the data could be arranged as in Figure 2.

One could argue that such structure can be accomplished in SAS by dividing the data into three separate data sets as in the segments above. Indeed, this is the only way that the data could be processed by procedures. To do this, we must duplicate certain key information in each data set in order to maintain the identity of the data source. This leads to redundancy; the same data being maintained in several places. The problem with redundancy is that unless the user is a very good data manager, data inconsistency will sooner or later crop up. This takes us into the second part of the definition of IMS.

A data base is not very useful without a manager. IMS manages data bases through its data manager called Data Language/I (DL/I). The functions performed by DL/I can be summarized as follows:

* Provides the facility for defining the physical data structure to IMS.
* Provides the facility for defining the structure with which an application program views the data.
* Provides the linkage facility to allow user programs to access the data.

IMS also provides facilities for managing the data bases through a network of terminals. This is the data communications part of IMS. The current implementation of SAS/IMS does not support IMS terminals; therefore, these facilities will be ignored in the remainder of this paper.

One major advantage of IMS is its data sharing capability. Multiple users can have access to the data without the usual data set contention typical of SAS and other OS data sets.

In summary, IMS is an environment. In it we can define data bases which serve as depositories for our data. It provides a manager for the data to help reduce data redundancy and promote consistency. It allows its users to access the data through a common interface which promotes data independence. And finally it allows data sharing.

What is SAS/IMS-DL/I?

The facilities that DL/I provides to access IMS data bases are available only to programs written in assembler, COBOL or PL/I. SAS/IMS extends that facility to programs written in the SAS language. The implementation of the interface was done using the INFILE/FILE as well as the INPUT/PUT statements of SAS[2].

Note that SAS/IMS deals only with one of the functions of the DL/I manager, the user-data linkage function. Before linking to IMS a data base must be established, and a view of the data base must be defined for SAS/IMS to work with.

How do I Get My Data into IMS?

The process of defining the data to IMS is called the Data Base Definition Generation (DBDGEN). The Data Base Definition (DBD) is a data module which contains information about the layout of the data in a data base. The module is created using assembler language macros statements. The following statements could be used to define our example data base STUDENTS:

```
DBD: STUDENTS
PRINT NOGEN
DBD NAME=STUDENTS,ACCESS=HIDAM
DATASET DDNAME=STUDDB,DEVICE=3380,SIZE=(4006)
SEGM NAME=STUDENT,PARENT=NULL,BYTES=100
FIELD NAME=(STUDID,SEQ,U),TYPE=C,START=1,BYTES=9
SEGM NAME=SEMESTER,PARENT=STUDENT,BYTES=50
FIELD NAME=(SEMID,SEQ,U),TYPE=C,START=1,BYTES=8
SEGM NAME=COURSE,PARENT=SEMESTER,BYTES=50
FIELD NAME=(COURSENO,SEQ,U),TYPE=C,START=1,BYTES=4
```

In the above DBD statements, we have defined the hierarchic structure of the data as discussed previously. In addition, we have specified which field of each segment will serve as the key. We can also define all the other fields in the segments, but this is the way the DBD is usually defined by Data Base Administrators.

Now that the DBD statements are done, the DBDGEN is accomplished as follows:

```
DBDGEN: STUDENTS
```

DBD Statements
Object Deck
Linkage Editor
Load Library (DBDLIB)
Once the DBDGEN is done, we must build a module called the Program Specification Block (PSB) using a similar process called the PSBGEN. The idea here is to define a data module that tells IMS what the program is to work with from the data base and what sort of activity the program is to effect upon the data. The primary purpose of a PSB is to allow the concept of logical data structures. This means that a program can be given access to only a portion of one data base, or even to portions or all segments of several data bases. But let's not be logical; let's simplify things by staying with the physical image of the data base.

A PSB will be needed to "start" the data base. This is called the "Data Base Load". The following statements define a PSB for that purpose:

```
PSB: STUDENTS

PRINT         NOGEN
PSBNAME=STULOAD
END
```

The PSB can then be built into a load library using the assembly-linkedit process described above for the DBDGEN.

While we are doing this, we can also define a separate PSB named STUDALL that we will need to process the data base after the initial load. For this PSB, the PROCOPT will be "A" which will allow us to add records, update or delete the data, and retrieve information as needed.

The next step is to define the actual data sets that are to contain the data. With a little planning, good estimates of the amount of space needed can be made and the data sets can then be defined or allocated. We have finally reached the point at which we can look at our SAS/IMS manual [3] for help.

How Do I Get My Data Out of IMS?

We have finally arrived at the purpose for SAS/IMS. Actually, SAS can be used to load the data base as well as for any other processing after that: inquiry, update or delete. Rather than go into a detailed explanation of how to process data bases using the SAS/IMS interface, we will leave that to the manuals and concentrate on the SAS/IMS to user interface.

Earlier we described a classification of users according to the type of work they do. Now we will discuss the difficulties that these users would have understanding SAS/IMS concepts and the efforts that would make these concepts easier to assimilate. The most important is the concept of navigation. By default, SAS processes data bases sequentially. Using the STUDENTS data base as an example; sequential means that with the first input statement execution we would retrieve the first STUDENT segment. The second input would retrieve the first SEMESTER segment for that student. The next series of inputs would retrieve COURSE segments for that semester. Then we would go up the hierarchy to the second semester, down again to retrieve its courses, etc., until done with the data for the first student. We would move then to the next student and similarly for all students in the data base. SAS/IMS detects the end of the data base and automatically stops input processing for that infile in the data step.

The first difficulty encountered is how to read the data into SAS variables. We can view the data base as a file of variable length records. Using a null input (INPUT @;) we can obtain the name of the segment being read and use it to conditionally execute the appropriate input statement defining its segment layout. The null input is necessary because the segments do not usually have a record type indicator within them.

The second difficulty is encountered when we try to reconstruct an observation from the information stored in the data base. The problem here is the variable nature of the number of occurrences of segments under a given parent segment. In our example, we can program for a finite number of semester segments, using SAS arrays, but the problem gets worse as each segment could have many courses recorded under it. An array would then be needed for each semester to accommodate the courses, etc. In summary, the fixed length nature of the SAS observation does not lend itself well for processing IMS observations.

A better alternative is to create relation-like data sets as we navigate the data base. The idea here is to break down the data base into several SAS data sets; one for each segment type. However, in order to be able to piece the segments together we must RETAIN in lower level segments the key variables of the hierarchical segments above it. For example, the data sets making up the STUDENTS data base would be:

```
DATA
  STUDENT  
    (KEEP=STUD_ID NAME CUM_CR CUM_AVG CUM_ETC)
  SEMESTER 
    (KEEP=STUD_ID SEM_ID SEM_CR SEM_AVG SEM_ETC)
  COURSE  
    (KEEP=STUD_ID SEM_ID CRS_ID CREDITS GRADE);
```

Using this method, as we recognize a segment being read, we would OUTPUT to the corresponding data set. Later, if necessary, MERGE statements could be used to combine data from two or more data sets.
Our experience tells us that the later approach is easier for research users to understand although it can be very inefficient if only a small subset of the database is needed for processing. Users in the programming support category are also better able to grasp SAS concepts using this approach. A small subset of administrative users are able to understand these techniques; the rest want an easier interface. Instructional users do not like IMS.

A second major concept used by IMS is the transaction. The IMS online system uses the transaction as a basis for database processing. The idea is to zero in on a subset of the data, do some processing with it, update segments if necessary, and report to the terminal user the results of the query or processing. For example, one could write a transaction to update, add or delete a course from a student's semester schedule. A screen is displayed to the terminal. The user enters the necessary key information to locate the record (STUD ID, SEM ID, CRS ID) and the option (ADD, UPDATE, DELETE). If an add is desired, the user enters the new course information. The transaction program processes the request and returns the screen to the terminal with the appropriate message.

The major difficulty in using transaction processing with SAS/IMS is the current lack of a full-screen facility in the data step. Transactions can be simulated using FSEDIT as the data collection mechanisms and one or more support data steps, but the cost of execution is usually large.

Final Comments

Prior to the announcement and release of SAS/IMS-DL/I, SAS users were asking for a SAS/DBMS. The availability of the IMS interface has not reduced the need for such a product.

SAS/IMS-DL/I extends the SAS language to support IMS data management functions. As stated in its manual, SAS/IMS is not an end-user product. However, if you understand the needs and capabilities of the user it is to serve, it can become a valuable tool in your organization.

A young product, SAS/IMS is constantly being nurtured by additions to the SAS product line. Features like the MACRO language, the TSO statement, and a growing family of full-screen facilities can help solidify its user base.

References


Figure 1. Data Set STUDENTS

<table>
<thead>
<tr>
<th>STUD-ID</th>
<th>NAME</th>
<th>COM-CR</th>
<th>COM-AVG</th>
<th>COM-ETC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEM1-CR</td>
<td>SEM1-AVG</td>
<td>SEM1-ETC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEM2-CR</td>
<td>SEM2-AVG</td>
<td>SEM2-ETC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEM9-CR</td>
<td>SEM9-AVG</td>
<td>SEM9-ETC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE1</th>
<th>CREDITS1</th>
<th>GRADE1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COURSE2</td>
<td>CREDITS2</td>
<td>GRADE2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COURSE9</td>
<td>CREDITS9</td>
<td>GRADE9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COURSE1</th>
<th>CREDITS1</th>
<th>GRADE1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COURSE2</td>
<td>CREDITS2</td>
<td>GRADE2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COURSE9</td>
<td>CREDITS9</td>
<td>GRADE9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. Data Base STUDENTS

STUDENT ADAMS
(CUM. INFO)

FALL SEM. INFO

SPRING SEM. INFO

SUMMER SEM. INFO

COURSE 1 INFO

COURSE 2 INFO

COURSE 3 INFO

COURSE 1 INFO

COURSE 2 INFO

COURSE 1 INFO

Root Segment

Semester Segment(s)

Course Segment(s)