I. Introduction

The Health Information Systems Office (HISO) of the Department of Health in the State of Hawaii is responsible for the design, development, and implementation for all data systems necessary to service Departmental programs. The range of Department services include mental health, alcohol and drug abuse, communicable diseases, developmental disabilities, maternal and child health, and vital statistics. The traditional approach to meeting these data needs has been the development of application specific batch software written in either PL/1 or COBOL. The result of this approach has been an ever increasing number of minimally effective data systems with unproficient user interfaces.

All Departments of the State Government utilize a central IBM 3013. While the Department of Health has advocated the use of interactive data base techniques in this environment, the realities of staffing and bureaucracy have thwarted any progress. The Department has advocated the creation of a Department of Health data center configured with an appropriate minicomputer and data base management software. A project to finance such a data system by a combination of grants and State funds is in progress. In the meantime, all Department of Health data applications are batch mode in a mainframe environment.

The Department received permission to install SAS on the State computer in 1978. The impact on retrieval problems was immediate. We were able to train personnel with minimal technical background to use SAS as a statistical report writer. The ease with which SAS can add new variables to SAS data sets and change variable lengths, formats, and labels made it potentially attractive as a vehicle for maintaining Departmental data bases in a batch environment. The decision was made, therefore, to investigate the use of SAS as foundation software in the creation of a generalized batch data base update system. (1) Generalized refers to the fact that the system is application independent. (2) Foundation, in this context, refers to the fact that even though PL/1 programs are components of this system, SAS is used to perform the actual update of a user master data base. (3) The term data base is defined in the SAS sense in that an application data base consists of a SAS data base with one or more SAS data sets as members. Each SAS data base is maintained as a generation data group on either magnetic tape or direct access storage.

GENSYS is currently operational in the Department's tuberculosis branch in which the SAS data base consists of one SAS data set containing patient registration data and another SAS data set containing service data. The system is expected to be operational in the Department's leprosy program within six months.

II. GENSYS Design Considerations

Several system characteristics emerged as essential in configuring a system which would permit the generalized update of a SAS data base.

A. Easily understood and unambiguous system transaction operation codes

B. An application independent transaction input format

C. Generalized data editing capabilities
   1. Flexible generalized batch data validation
   2. Application specific editing

D. An application independent SAS update routine to accomplish the actual update function

E. A concise informative update report.

The solution to these problems evolved into a system (GENSYS) which consists of four PL/1 programs in concert with SAS. A schematic of GENSYS is shown in Figures 1 & 2. The system is used in two phases:

1. Create and load an application specific dictionary (one for each SAS data set comprising a data base). See Figure 1.
2. Update an application data base. The programs are executed in the sequence indicated in Figure 2 to perform an update. See Figure 2.

Easily understood system capabilities were provided by defining three transaction operation codes for updating data bases.

(1) A: Add an observation to a data set as a function of user specified key(s).
(2) C: Change (update) any variable in an observation other than the key(s).
(3) D: Delete an observation. (Deletes of non-key variables are accomplished as changes in which the variables are set to "missing" in the SAS sense).

An application independent transaction input format was achieved by adopting a convention similar to named input in SAS. Each variable is associated with a numeric tag. For example, the variable "SEX" could be associated with tag 03 and the variable "DIAGNOSIS" with tag 15. Tag 00 is reserved for association with the transaction operation code and tag 01 for association with the primary key of the SAS data set. Each tag is preceded by a special character (break character) and followed by the actual variable data. For data sets containing less than one hundred variables per observation, the break character is the logical or-bar (12-7-8 punch). Transactions are prepared as a stream (in the PL/1 sense) of characters consisting of break characters, tags, and data. The general model is:
A variable and its tag are ignored if not present on the data collection form.

(2) A, C or D is the transaction operation code.

(3) The data following the operation code is the primary key.

(4) Variables may be entered in any order within a transaction.

(5) The data may be preceded and followed by as many blanks as desired.

(6) An asterisk is punched following the tag to delete a non-key variable.

Generalized data edit capabilities are accomplished by a construct known as the Data Dictionary. While the main function of the Data Dictionary is to validate (Program 3 in Figure 2), Program 2 and Program 4 access variable length information from the Dictionary in order to support reformat and error print functions. The Dictionary is not used by SAS.

The Data Dictionary is a partitioned data set, each member of which contains information describing the variables of a particular SAS data set. The members of the Data Dictionary are essentially array structures and are accessed by the PL/I programs using the locate mode input technique. The variable tags described above are the array indices. A dictionary contains editing and length information for each variable. Note that the versatility of SAS program statements allows for the option of creating new variables whenever an observation is added to a data set. These variables, created by SAS via program statements at update time, are not members of the Data Dictionary since such variables are formed by program statements consisting of calculations involving transaction variables.

III. Functional Description of GENSYS Software

Program 1 - Load Data Dictionary

Using a concise Data Dictionary language, the user loads a variable name, variable length, and editing patterns (or) for each variable. Currently supported edit patterns are:

(1) Data not to exceed a specified maximum length
(2) Data must meet a precise length specification
(3) All characters of data must be numeric
(4) Data must be in format of a PL/I decimal or decimal floating constant
(5) Data must meet a specific numeric range condition
(6) Data must assume a value from a specified numeric list of constants
(7) Data must be in format of a character bit string (i.e. all characters are '1' or '0')

(8) Data must be a valid date (Formats are YMD, MDY, YMMD, MDYAY, YMMDAY, AYMD, MMDA)
(9) Data cannot be updated to missing (in the SAS sense)
(10) RATS edit. RATS is an acronym for RAPID ACCESS TABLE SYSTEM. RATS is an assembly language routine which is link edited with Program 3 and allows for the capability of table look-up. The data element is treated as an index to a RATS table indicated in the Data Dictionary. Options allow either for the determination that the argument is present in the specified table or for the replacement of the argument by a RATS returned table value.

An example of the dictionary specifications for a hypothetical field follows:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELE.E.EDT(05) = 3</td>
<td>ELE.E.NAM(05) = 'COUNTRY'</td>
</tr>
<tr>
<td>ELE.E.EDT(05,1) = 2</td>
<td>ELE.E.EDT(05,2) = 3</td>
</tr>
</tbody>
</table>

The statements define a variable with a tag value 05. The variable is assigned a label 'COUNTRY', has a length of 3, and is to have edit patterns 2 and 3 sequentially applied by the batch data validator. The variable names in the Dictionary correspond to the variable labels in a SAS data set.

Program 2 - Reformat Transactions

As noted above, the selection of a break character in the transaction input format is a function of the number of variables in a SAS data set. Program 2 reformats all transactions to one fixed format after determining which break character is used.

Program 3 - Batch Data Validator

Transactions are input in the format described above and transformed to the format of SAS named input (i.e. label-data where label is the three character numeric tag described above). Each transformed transaction contains an operation code (A, C, or D) and a system variable used by Program 4 in order to sort error messages. A transaction is not passed to SAS if:

(1) The transaction operation code is invalid;
(2) A key variable is not present in a transaction; or
(3) A key variable fails an edit pattern.

Appropriate error messages are written to the error file for all error conditions. Each variable is edited according to edit patterns stored in the supporting Data Dictionary member. An appropriate error message is written to the temporary error file for each instance of a variable failing an edit pattern. A variable is not included in a transaction passed to SAS if it fails any of its edit patterns.
The SAS Update Program

The UPDATE function of SAS is used to update each data set of a data base. After the update cycle, the additional data sets of the data base are copied (PROC COPY) to the new generation of the data base in order to ensure that a reference to the latest generation of a data base includes all member data sets.

Transactions passed from the batch data validator are processed by SAS as named input into the SAS data set TRANS. TRANS is then sorted into the operation code order D, A, C. This sort order allows for the deletion, addition, and possible update of an observation within a BY-group.

The update is accomplished by utilizing the SAS update capability in conjunction with a SAS subroutine. The SAS update capability is driven by the statements:

```
DATA NEWSTR.DAT; UPDATE CLDMSTR.DATA TRANS; BY VAR1...VARN;
```

where

1. NEWSTR is the DD statement describing the new generation of the SAS data base.
2. CLDMSTR is the DD statement describing the current generation of the SAS data base.
3. DATA is the name of the old master and new master SAS data set.
4. TRANS is the data set of transactions passed from the batch validator.
5. VAR1...VARN are the update keys.

The SAS subroutine is invoked when the last transaction observation with a BY-group has been processed.

The main function of this routine is to detect errors and produce error messages for the following application independent transaction error conditions:

1. ADDITION REQUESTED, MASTER EXISTS
2. ADDITION REQUESTED, CONFLICTING TRANSACTIONS
3. UPDATE REQUESTED, MASTER DOES NOT EXIST
4. DELETION REQUESTED, MASTER DOES NOT EXIST
5. DELETION REQUESTED, CONFLICTING TRANSACTIONS

For all error conditions, the master observation is left intact if the transaction is in error. For example, if the generalized update routine detects that the user is attempting to add an observation and the transaction key(s) are existent in the master data set (error condition 1 above), then the transaction is deleted and the master observation is left in place. This routine solves a problem with the example of the classic update displayed in the SAS APPLICATION GUIDE. In the classic update example, the problem of attempting to add an existing observation is addressed by simply displaying the resulting erroneously "updated" observation with a PUT ALL statement.

The update routine relies heavily on the powerful SAS array capabilities. Variables are maintained in the master data set with their SAS labels which correspond to the labels stored in the PL/1 Data Dictionary. Variables in the SAS data set TRANS are maintained with the tags described above as labels. This technique permits the definition of an array of master variables and an array of corresponding transaction variables. When a transaction is accepted for update, master variables are replaced by their analogs in the transaction data set. TRANS variables are then dropped by using a SAS DROP statement.

The concept of CONFLICTING TRANSACTIONS used above arises from the convention of the SAS UPDATE that even though several observations from the transaction file can have the same BY-group, SAS outputs an observation only after all transactions are applied. This convention, in concert with the transaction sort order of GENSYS, allows for some troublesome sequences of transactions within a BY-group. One example is detailed. It is possible that the user could submit for update within a BY-group a sequence of codes such as A, A, C, C, C. This sequence would be defined as conflicting and, hence, in error since the user is requesting a multiple add of the same observation within one BY-group.

All application specific editing is stored as SAS program statements in the SAS macro library. Application specific editing is performed after the global editing for transaction integrity described above in this section. Application specific editing tests associative relationships between the variables of a transaction or between the variables of a transaction and its targeted master. For example, SAS program statements could stipulate the rejection of an add transaction if the transaction variable SEX='MALE' but the master variable OCCUPATION='HOUSEWIFE'. At system implementation time, the user has complete freedom to define SAS program statements for inclusion in the SAS macro library. A transaction failing an associative edit causes either a warning or the rejection of the transaction. For those applications in which the SAS data sets of a data base are not independent, more complex associative editing is accomplished by MERGING the TRANS data set with one or more members of the data base in order to extend the TRANS data set with additional variables. Such extension variables can be used either for editing purposes only or can be treated as variables to be updated into the new master data set.

Program 4 - Print Update Report

Program 4 reads the error and message file written by programs and the SAS update program and prints the update report. A unique feature of the update log is that it displays, for each transaction, a detail of both the data value reflected in each transaction and any impacted master observation data.

For example, the user might specify an update transaction as:

```
PROC 12345678902 203 MAIN STREET 11 BLUE.
```
Assuming that the Data Dictionary has defined tag 01 as the primary key (in this example SOCIAL SECURITY NUMBER), tag 02 as the variable SEX, tag 03 as ADDRESS and tag 11 as COLOR, then the format of the message on the error log for this transaction would be:

```
&001123456789&0022&001123456789
SOC SECURITY 123456789 123456789
SEX 1 2
ADDRESS BROOKLYN MAIN STREET
COLOR BROWN BLUE
```

The first line is simply the reformatted transaction from Program 2; the second line highlights the primary key and the operation code; the remaining lines display the field name, the old master data and the new master data for each variable participating in the transaction.

In addition, the program displays the number of observations in the old master, the number of observations in the new master, and sums the number of accepted and rejected operation codes by operation code type.

IV. Conclusion

GENSYS is proving to be a valuable program in the Department of Health in that redundant programming is being avoided. The batch data validation appears to be a solution to the problem of coding edits which are independent of the application. The versatility of SAS program statements in a macro library to support associative editing is a particularly useful feature of the system. The versatility of SAS in adding new variables to existing SAS data sets makes it attractive as foundation software in an updating system. Data systems in the public health environment are volatile in structure because of always changing State and Federal reporting requirements. The authors hope to investigate creating a model of the system which would support a hierarchical relationship among differing observation types in one SAS data set.