Beyond Field Validation:
Incorporating the Batch Edit Into the Total Data System

John Quarantillo, Westat Inc., Rockville, MD
Judy Rayner, Westat Inc., Rockville, MD

ABSTRACT

Computer Systems professionals have always been faced with the problem of 'dirty data'. In the past, errors were detected using batch edit programs to identify errors. The output from these batch edits was then reviewed, corrections were transcribed onto coding sheets, then keyed, and finally applied to the data using a batch update program. With the advent of custom interactive data entry systems, many edit checks were built into the data entry system, replacing the batch edit process. There are clear advantages to this, but in the interest of speed and simplicity, the data entry edits were not as robust or thorough as the batch edits of the past. Many complex checks were left undone. The approach that we describe uses the SAS® system, and SAS/AF® to combine a customized interactive data entry system with improved batch systems, taking advantage of the strengths of each to produce an integrated system for data entry and cleaning that is both thorough and easy to use. Methodologies such as those outlined here are important additions to the front end of any data warehousing effort. Our system is used under, but not limited to the MVS and Windows environments.

INTRODUCTION

In the research environment, it is often difficult to have complete control over the quality of the data that are collected. This can be due to errors in the original data, illegibility of the initial documents, a failure to consult all available materials at the time of data collection, judgment and/or careless errors on the part of the data abstractors, and keying errors. These errors have traditionally been detected using a computer program that produces a hard copy listing of the potential errors on which clerical staff note the corrections. These corrections are then keyed and used to update the data. This process is repeated until the edit output contains no errors, or only errors classified as "overrides" because for some reason they cannot or should not be corrected. At this point the data are as "clean" as can be achieved with reasonable effort.

Although this methodology has successfully helped in the resolution of errors for a number of years, it does have a number of drawbacks, which, until recently, were accepted as unavoidable:

- errors can occur in both the abstracting step and the keying step during data entry,
- while editing, there is no easy way to obtain information about the keyed data that were not printed on the edit output,
- additional errors can be introduced during both the recording and keying of update records,
- the revised record cannot be viewed at the time the edit decision is made,
- statistics about the nature of the errors are not easily obtained,
- the update process seems excessively involved when correcting a small number of errors,
- the same overrides must be reviewed on every pass of the edit, and
- the update process is rather time consuming.

The project requirements for this application were that it address the weaknesses listed for the existing edit methodology without adding any new limitations. The software needed to be easy to use for non-programmers, and it had to run on both the IBM mainframe and the PC with few apparent differences to the user. There were insufficient funds to develop two applications, so we needed to be able to develop one product that would run in both environments. We wanted to be able to use the same software both for data in a large, expanding, multi-user data warehouse and also in a separate, smaller, single-study application. The final product would also be integrated into a larger, interactive system that provides data input, querying, reporting, and analysis support. SAS® software seems to offer one of the best environments today in which to develop such versatile applications.

DESIGN OF REVISED PROCESS

We decided to use SAS/AF® software to create the data entry and edit modules, because it provided the needed interactive support, offered greater power and flexibility than SAS/FSP® software procedures such as FSEDIT, and would run within the specified hardware configurations.

Data Entry Component

This component of the system included application specific interactive data entry whenever practical. By using the data entry component, the abstracting step was often eliminated. Table lookups, pop-up selection lists, and other real time data validation methods were used to eliminate most keypunch errors. Many errors that would have been introduced at the time of abstracting the hard copy data were also eliminated. Examples of this type of error are invalid codes, fields out of range, etc. The user is notified of such errors at the time of data entry, and the errors are corrected before committing the record to the database. However, edits for more complex errors were not included in the data entry software. Attempting to make the data entry system 'foolproof' would cause the data entry component to be slow and cumbersome. The more involved edit checks were included in the batch edit that is to be run separately from the data entry process. There are also times when data will not be entered using the system data entry software either because it is already in electronic format or an abstract is just more practical. In these cases a full edit program will be needed. Either way it is apparent that the improved modes of data entry do not eliminate the need for some editing support.

Batch Edit Component

Using SAS/AF allowed the batch edit to be incorporated easily into the overall system. The batch edit itself can be written using base SAS software. Existing batch edit programs could be incorporated into the system. Our application required a more complex batch edit that traversed the relational database for each logical record, so the batch edit was written using SAS/AF Screen Control Language. An interface was developed to create a menu driven environment that was simple to use. From the user's point of view the following steps are followed to edit a data file.

1. **Run the batch edit.** A menu option is selected to run the comprehensive edit on the file. The first time the batch edit program is run on a given data file, it creates an error file containing information about the errors detected in the file and initializes an archival error file. It also creates a hard copy listing, which optionally may be printed, displaying the records with errors and printed information about each error.

2. **Review the hard copy output.** This step is optional, but is highly recommended for large or complex data files. This step involves taking the hard copy printout of the errors and using it to resolve all of the more complicated errors. This listing can be taken to data experts, original data sources, a convenient work space, or other locations for resolving errors. The listing offers a convenient place for jotting notes on how to resolve the errors.

3. **Interactively classify and correct errors.** The error corrections and error types are now entered into the computer. If the data are simple, step 2 can be skipped, and the person entering the corrections can sit down at the computer and begin this step as soon as the batch edit has completed running. The error review/update process is facilitated by an error correction module that displays on the screen each error detected by the batch edit. The user has the option of processing the errors sequentially or directly by searching for specific errors. The user classifies the error by type, and the program stores this code in the error file. The user is also permitted to skip errors and review them at a later time. For our applications we have five valid error types:

    - **Override.** An override is a possible error that has been found by the batch edit, but that for some reason cannot or should not be corrected. If, during the review/update process an error is classified as an override, that classification is stored in the error file. Any subsequent runs of the batch edit program checks any errors found against the error file. If an error has been previously classified as an override, that error is not included in the batch edit output. The override status is removed if the value for the field is, at some later point, changed. This eliminates the re-reviewing of previously identified overrides.

    - **Pending.** A user would classify an error as pending if the correct resolution cannot be determined during this edit cycle, but a correction is expected some time in the future. The next pass of the edit will not execute until an error type has been entered for every error.

    - **Keypunch**, **Abstracting**, or **Coding Error.** For all other error types, the user is taken directly to the data entry screen containing the problem field(s). The user can then correct the error. Because the update program uses the original data entry software, any of the simple edits included in the data entry software are enforced at the time of error correction, decreasing the likelihood of introducing additional errors. The different classifications distinguish between the perceived sources of the error. Clients often request statistics on error rates for particular sources, so this type of data will facilitate the reporting of these statistics.

4. **Produce error summary report.** The program supports the display and printing of error summary reports. These can be custom designed for the application and will usually include tabulations on error codes and variable names.

5. **Repeat the process until the data are clean.** After a code has been entered for all detected errors, step 1 should be repeated. However, for all subsequent passes of the batch edit, the actual processing is somewhat different. The edit program first appends the error file which now includes error type codes to the error archival file and re-initializes the error file for the upcoming set of detected errors. If an error is detected during a subsequent edit pass, it is compared against the error archival file to determine whether the same error has been detected previously and, if so, what error code it received. Errors that have already been overridden are not reported again. Otherwise the steps are the same as in the first pass.

The manner in which the records are accessed for error correction was designed for quick movement to the most likely problem area on the record for the error being processed. In many research studies, a considerable amount of related data are stored in relational databases, where data for a single entity, such as a person or an exposure survey, are contained in as many as four or more related data sets. Sometimes three or four data entry screens are required to display the data from just one record in one of these files. Therefore, it can take a noticeable amount of time to search for the correct record in the correct file and then page down to the correct screen. Taking the user directly to the most likely problem area often decreased access time by a factor of four or even more.

One of the goals of our system was to develop something that the users would find somewhat familiar, but better. The revised system, as described here, is still based on the batch edit. This strategy was chosen for a number of reasons. Some of the logic checks that get requested involve multiple files and between-record checks that can be time consuming and inefficient when implemented interactively. When programming a collection of such checks, it is often most efficient to pass through the data set multiple times using different orders and relationships for different checks. However, most users prefer to see all the errors for a given entity together so the hard copy records only need to be pulled once. In addition, the order in which they prefer the error messages to be presented for a given entity is often different from the order in which the errors are identified. As a result there are many reasons why the order in which errors are discovered is not the order in which they should be presented. Such reordering is best handled by a batch job. The printed output is often a large aid in resolving the errors because it can be taken to the place or person with the solution, provides a convenient place to jot relevant notes, and can serve as a written record of the decisions. Therefore a batch edit avoids tying up the system while

complicated checks are conducted, and it facilitates the creation of a better error display layout and order.

ADVANTAGES OF REVISED PROCESS

By combining the data entry system with the batch edit, the end product proved to be as powerful as it was useful. Listed below are the objectives met by the combined approach:

1. **Eliminates Abstracting.** The implementation of the interactive data entry system using SAS/AF allows the users to perform data entry directly from hard copy. All coding of fields is handled using selection lists. Therefore, errors or problems that would be introduced during abstracting are avoided.

2. **Interactive Data Entry System Reduces Errors.** Many coding, abstracting, and keypunch errors are eliminated by the implementation of edits during the data entry process.

3. **Amount of Data Keyed for Updates is Reduced.** All the information identifying the potential error, its record, and its field(s) are stored in the error file. This information is used by the review/update program to zoom the user directly to the proper place to update and correct the record. Therefore the user is only keying the update information, not any of the identifying information necessary to apply the update to the correct field. Problems such as the miskeying of record and/or field identifiers are minimized.

4. **Speed of the Update Process is Increased.** The update process speed is greatly increased by not requiring the user to navigate to each offending field before making the updates.

5. **Number of Update Steps is Reduced.** By combining the recording, keying, and actual updating into one step, the coder makes the correction directly to the problem record.

6. **Subsequent Errors are Reduced.** By using the lookup tables and other features of the interactive data entry system, the data entry edits are applied to the updates at the time of the error review/update process. Potential new errors are caught before being introduced.

7. **Updates are Done in Context.** It is often difficult to discern the correct value for a field without the knowledge of the contents of other related fields. The update interface connects you directly to the data entry system, giving you the ability to review all the fields on a record and even other records, prior to keying the update. After keying the update(s), you see the revised record in context, allowing you to see if the correction is appropriate.

8. **Re-review of Overrides is Eliminated.** The batch edit must be run repeatedly after each update session until no resolvable errors are identified. It is wasteful to re-review errors previously judged to be overrides. Once an error is identified as an override, that error is suppressed on all subsequent batch edit runs.

9. **Reports on Errors and Solutions.** The introduction of the error file greatly facilitates reporting on the batch edit process. Descriptive information regarding number, type, and source of errors can easily be presented. Information gained from these reports will allow data managers to possibly revise the edit rules. The table will also make it easier to provide error rates that clients and project staff might request.

CONCLUSION

This system combines an old methodology, the batch edit, with some of the latest advances found in interactive data entry systems often built into the data warehousing approach. The resulting product allows experienced data editors to have the best of both worlds. They can submit batch edit programs similar to ones they have used for years. They can obtain the same (or better) hard copy output that they have found so useful in the past. But now they can quickly and easily record error status codes and update the data. Overrides no longer need to be reviewed over and over again with each edit pass. Clients and managers will be happier with more informative and accurate error status reports. By integrating the old and new methodologies and incorporating the result into a data warehouse type system, a useful and successful product has been achieved.

ACKNOWLEDGMENTS

We would like to acknowledge the following:

- David Utterback of NIOSH and Kathy Fraeman of Westat Inc. for assistance in systems design.
- Savita Marathe of Westat Inc. for programming the system.

For Further Information Contact:

John Quarantillo
Westat Inc.
1650 Research Blvd. Rm. WB236
Rockville, MD 20850

quaranj1@westat.com