RETRO-ENGINEERING A SAS® REPORTING SYSTEM FRONTEND

Brent Turner, City of New York

INTRODUCTION

The Financial Information Services Agency (FISA) of the City of New York has been created as a requirement for federal subsidies used to assist the City in meeting its financial obligations. These subsidies had initially been requested by the City in response to its fiscal crisis in the mid-1970's. FISA has two principal functions, which are:

- to provide financial information enabling municipal authorities to budget for short- and long-term capital expenditures
- to manage the payroll for the multiple agencies of the City of New York.

To fulfill these functions, FISA employs computing hardware which includes an IBM 3084, running in partitioned mode, and an IBM 3081, both using the MVS/XA operating system.

Payroll processing is performed by two separate software systems which each handle different city agencies. The systems are Phase II and Payroll Management System (PMS). Phase II is a collection of independent programs, each of which separately processes for the agencies that are still serviced under it the payroll for different groups of agencies. However, the processing is not done in a uniform manner nor at the same time. PMS, on the other hand, is designed as an integrated system, with a central set of tables and a timesharing method. It is intended to eventually replace the older Phase II through the systematic migration of the remaining agencies in Phase II to PMS.

PMS is a massive IMS/VS online and batch system consisting of COBOL and assembler programs that access DL/I data bases. The biweekly checks for the agencies served by PMS are generated by a complex system of batch jobs called PAYCYCLE, which is run every other weekend. As a result, there is a very narrowly defined timeframe to run the PMS payroll process with almost no margin for error. It has become increasingly difficult to meet this time window as more of the city's agencies (and thus, more employees) have been migrated from Phase II to PMS. Time overruns have started to occur as a consequence of this.

Included in PAYCYCLE processing is a series of twelve SAS programs which had each accessed a sequential file produced by PMS, called the Pay Cycle Extract File (PCEF). PCEF is a comprehensive repository for all employee data pertaining to payroll processing. The SAS programs each report upon some aspect of PCEF which is used to monitor the outcome of the PAYCYCLE. A preliminary examination of these programs had revealed opportunities to reduce the processing time, which had been over fifteen hours, by more than 50 percent.

THE PROBLEM

The original PCEF SAS report programs had not been designed as a system; further, they had neither been written by the same person nor even at the same time. Each program had been written as a standalone entity. As such, each had read PCEF to individually extract the particular data that it needed. This had not been a problem when the number of programs reading PCEF and number of employees processed through PMS had each been small.

As programs have been added to the PCEF reporting system and as the number of records to be processed has increased, a minor crisis has developed. With each generation now created by PAYCYCLE, the PCEF contains over three million records. It requires over 600 seconds (ten minutes) CPU time for SAS to read PCEF, and therefore had required over two hours CPU time for all the SAS programs to just read the data. This process had performed over 1.1 million input/output (I/O), i.e., data-read, operations, during which there had been contention with other programs that also read PCEF. Consequently, the total elapsed (wall clock) time for the SAS report process had been over fifteen hours.

THE SOLUTION

The first objective has been to reduce the number of I/O operations performed on the PCEF data by the SAS programs. Each PCEF record is read only once and, if appropriate, is then written to one or more SAS data sets. The data are then available to all the SAS report programs.
which are not then competing with other programs to read PCEF.

The observation-selection criteria for each SAS program (defined with suitable IF-THEN statements) have been incorporated into a PCEF-subsetting frontend program. This program consists of a single data step which reads the PCEF data to subset the file into thirteen SAS data sets. These data sets have been found, by inspection of the original programs, to be required for subsequent report processing. Consequently, each of the original programs has been modified to read only the appropriate SAS data subset(s) rather than PCEF itself.

Additional refinements have included defining only the data set variables which are used by the report programs. For this purpose the at-sign (@) pointer control facility is used on the INPUT statements. This replaces the prior method of defining all the fields for each PCEF record.

Further, the SAS data sets are each written to only one of three SAS data libraries, which themselves reside on different disk drives. The segregated libraries enable most of these programs to be run concurrently instead of consecutively. This is due to the reduction of disk access-arm contention between the individual SAS report programs.

The separate operations of the frontend program are isolated into discrete modules using SAS macros. The program structure is a quasi-hierarchical design where process-initiation is controlled from a single macro named, appropriately, PCEF (see Figure 1).

It is quasi-hierarchical because three of the macros can invoke the same subordinate macro. These invoking macros are ATTRIB, DEDUCT, and DETAIL and they contain the INPUT statements for the employee attributes, payroll deductions, and pay detail records, respectively. The invoked macro contains the INPUT statements for variables which are common to all of these records, and consequently is named COMMON.

In addition, both the DEDUCT and DETAIL macros call their own separate subordinate macros, in true hierarchical fashion. These subordinate macros contain the record selection criteria for the thirteen data sets which are created; however, not all of the record types are required in each data set. Eleven of the macros select deduction records for eleven of the data sets; these macros are prefixed with DEDT. Six of the macros select pay detail records for six data sets and are prefixed with DETL.

The macros which are subordinate to the PCEF control macro are grouped into three categories according to the function which they perform. These functions are:

- variables inclusion, for each data set (thirteen macros, name prefix: KEEP);
- record layout specification, for different record types (five macros: HEADER, ATTRIB, DEDUCT, DETAIL, COMMON);
- record selection, for the different record types required by each subsequent report program (name prefixes: eleven DEDT, six DETL).

Instructions within the program documentation provide a methodology for the insertion of new modules at such time when it becomes necessary to use new record types and/or to produce new data sets for additional reports. Comments indicating the positions for such insertions are also provided within the program code.

RESULTS

By storing the selected records in permanent SAS data sets the total number of I/O operations is reduced from approximately 1.2 million to 270,000. This translates to a reduction in the number of records read from over 36 million to about 4.3 million. The latter figure includes the three million records read to convert PCEF to SAS-readable form (i.e., to SASSinate the data), and 1.3 million observations read by the report programs. This in turn decreases the CPU time, from over three hours to approximately 40 minutes.

Finally, the running time for the PCEF report process has decreased from over fifteen hours to five hours. This is due in part to the elimination of data-access contention with other programs attempting to read PCEF concurrently with the SAS programs. However, there is still some contention between the SAS programs attempting to read the SAS-format PCEF data, which indicates an area for possible additional fine-tuning. Nevertheless, a 65 percent-plus reduction in elapsed time is substantial for the initial effort.
The comparative differences in resource usage for both the non-integrated and integrated methodologies can be seen in Figure 2. Figure 3 presents the differences in both CPU and elapsed times for each PCEF SAS program both prior to and following the implementation of the frontend program.

**DISCUSSION**

The change in methodologies has resulted in significant improvements for the PAYCYCLE process. Among these has been the relief in the time constraints on scheduling, due to the reductions in required processing for the SAS PCEF report programs. These reductions have been attained through judicious use of computing resources from consideration of resource characteristics.

The principal modification has been to read PCEF only once, to convert the data to SAS-readable format. This change eliminates contention resulting from different programs attempting to read the same external file concurrently. Contention (and processing time) is further reduced by distributing different SAS data subsets across multiple volumes. This facilitates concurrent execution of the various programs attempting to read "SASified" data.

By keeping only the data that are genuinely needed for report processing, the number of subsequent I/O operations is reduced. To achieve this goal requires not only selecting the appropriate records, with IF..THEN statements, but also defining only those variables that are actually used. The latter objective is attained with at-sign pointer control on the INPUT statement.

The human programmer who is responsible for the report system must also be considered when such alterations are being planned and implemented. Program design determines the ease with which the programmer both develops and subsequently maintains any such system. Human beings exhibit faster and more thorough comprehension of structured designs where distinct functions are kept separate from each other.

Fortunately, SAS macro language provides the capability to implement a structured design, with an advantage for processing performance. The SAS macro processor generates program statements that appear to be inline code to the SAS supervisor. Inline code is much more efficient to compile and execute than external subprogram modules. This is because inline code requires neither the storing and loading of registers nor the branching within a computer address space as do subprograms in other programming languages. Thus, both human and machine are optimally served according to their different operational needs. By using macros to isolate the functionally discrete components of the frontend program, maintainability is achieved and extensions to the program are facilitated.

**ACKNOWLEDGEMENTS**

The author expresses his appreciation to Messrs. Devin Alford, Mikhail Rakmilavich, and Abraham Persky for their numerous contributions which facilitated the implementation of the integrated frontend SAS program itself.

In addition, the author expresses his gratitude to Dr. Hulan E. Jack and Mr. Thomas Fazio for their valuable comments and assistance, and also for their patience in dealing with the perturbations of the author, during the course of this transcription. For further information contact:

Brent Turner
F.I.S.A./City of New York
111 Eighth Avenue
New York, NY 10011
212/206-3115

SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.
Figure 1. Hierarchy of Program Macros: Relation of invoking to invoked is top to bottom.
### Figure 2. Comparative Resource Usage: Before and After Integration of Data-Reading

<table>
<thead>
<tr>
<th>PCEF Program</th>
<th>CPU Time Before</th>
<th>CPU Time After</th>
<th>Elapsed Time Before</th>
<th>Elapsed Time After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front End</td>
<td>N/A</td>
<td>00:20:27</td>
<td>N/A</td>
<td>02:43:39</td>
</tr>
<tr>
<td>Program 1</td>
<td>00:13:09</td>
<td>00:00:23</td>
<td>00:47:55</td>
<td>00:03:41</td>
</tr>
<tr>
<td>Program 2</td>
<td>00:13:31</td>
<td>00:00:22</td>
<td>01:03:32</td>
<td>00:02:08</td>
</tr>
<tr>
<td>Program 3</td>
<td>00:12:58</td>
<td>00:00:20</td>
<td>01:09:08</td>
<td>00:02:59</td>
</tr>
<tr>
<td>Program 4</td>
<td>00:12:53</td>
<td>00:00:17</td>
<td>01:02:49</td>
<td>00:02:15</td>
</tr>
<tr>
<td>Program 5</td>
<td>00:05:35</td>
<td>00:00:39</td>
<td>00:58:56</td>
<td>00:06:11</td>
</tr>
<tr>
<td>Program 6</td>
<td>00:23:56</td>
<td>00:00:33</td>
<td>01:44:18</td>
<td>00:11:14</td>
</tr>
<tr>
<td>Program 7</td>
<td>00:31:54</td>
<td>00:10:55</td>
<td>02:01:33</td>
<td>01:03:25</td>
</tr>
<tr>
<td>Program 8</td>
<td>00:24:26</td>
<td>00:02:41</td>
<td>02:09:45</td>
<td>00:20:55</td>
</tr>
<tr>
<td>Program 9</td>
<td>00:14:09</td>
<td>00:01:08</td>
<td>01:11:33</td>
<td>00:07:07</td>
</tr>
<tr>
<td>Program 10</td>
<td>00:12:58</td>
<td>00:00:17</td>
<td>00:56:15</td>
<td>00:01:32</td>
</tr>
<tr>
<td>Program 11</td>
<td>00:13:15</td>
<td>00:00:54</td>
<td>01:30:02</td>
<td>00:06:57</td>
</tr>
<tr>
<td>Program 12</td>
<td>00:12:48</td>
<td>00:00:26</td>
<td>00:50:16</td>
<td>00:03:11</td>
</tr>
<tr>
<td>Total Time</td>
<td>03:11:50</td>
<td>00:39:26</td>
<td>15:31:07</td>
<td>05:03:20</td>
</tr>
</tbody>
</table>

**Figure 3. Comparison of PCEF Program Processing Times: Before and After Frontend Implementation (in hh:mm:ss)**