Paper 168-26

“Just the facts ma’am”, meets “Is it safe?”

David Johnson, DKV-J Consultancies, Holmeswood, England

ABSTRACT

As larger data sets are processed into Data Warehouses, the time to build and analyse the data is increased. The information provided by the data is needed earlier so decisions can be made speedily to respond to issues or react to market forces.

The faster delivery times means load processes are tuned for speed, and interventions in the process are removed. But the long run times also mean rework time can impact business delivery.

INTRODUCTION

Consequently, batch processes need to have both detailed logs and data analyses as well as a ‘go/no go’ indicator on the batch process. The ‘go/no go’ indicator can provide the business with a confidence indicator on what will be loaded and the batch process with a signal to proceed with data update / indexing or data base consistency checking.

If there is any question about the data, then we need a more comprehensive analysis of the data structure. We may not have the time to run data tests on our Warehouse, so our batch process should routinely provide tabulations. Access to similar tabulations from previous months provides this comparison.
Our ‘go/no go’ indicator should be easily accessed, so providing this response through email is discussed. Finally, surfacing more detailed data through HTML on a corporate Intranet is demonstrated. This provides both the SAS session log in a manner that highlights errors and warnings, and the extensive SAS output in an easily indexed and accessible format.

OVERVIEW OF THE PROCESS

A mainframe production system is processed at each month end for new and changed records in its Policy and Claims registers. The records extracted are to become part of a Management Information System (MIS) that currently holds more than 12GB of policy transactions split between two partitions of the transaction data. About the same volume of data is spread across the Policy, Claim, Claim Transaction and code table areas.

The mainframe extract files are generated from programs written in Cobol, then cleansed and restructured in SAS processes. The SAS data sets are transferred to a Unix platform where further cleansing and restructuring is performed before the data is loaded to Sybase transfer tables. Then the Sybase transfer tables are loaded to a Sybase production system. They are then indexed and checked for consistency and finally released to the MIS user front end (Business Objects).

The process run times span the intervals shown in the diagram at figure 1.

The mainframe process is run on a production scheduler.

The Unix Development processes run under a 'bespoke' SAS scheduler that tests for file availability and initiates the SAS processes as soon as all data is available.

The Unix production process is currently initiated manually, although the process has been modelled to allow fully automated Sybase processes.

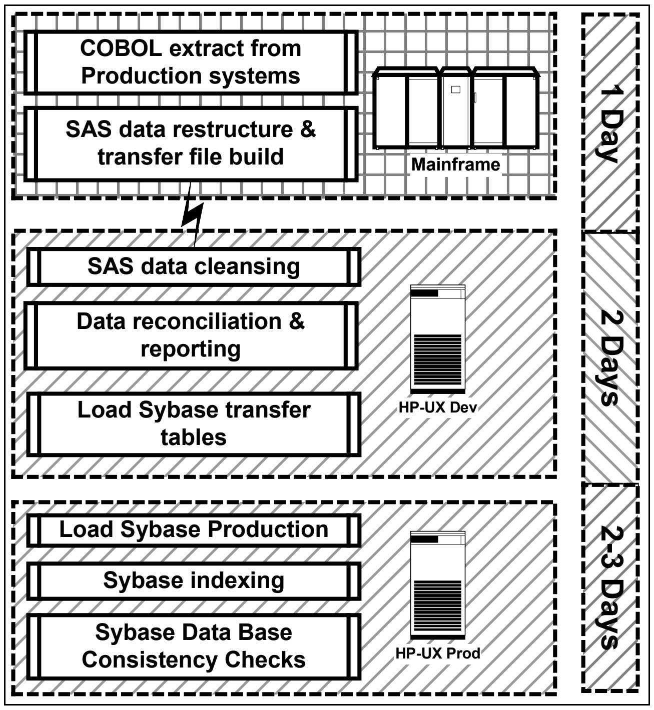
COBOL extract from Production systems
SAS data restructure & transfer file build
Mainframe
1 Day
SAS data cleansing
Data reconciliation & reporting
Load Sybase transfer tables
HP-UX Dev
2 Days
Load Sybase Production
Sybase indexing
Sybase Data Base Consistency Checks
HP-UX Prod
2-3 Days

Figure 1

This modelling involved providing a 'flag file' to the Unix job scheduler reporting SAS data was available and suitable for processing. For obvious reasons, it was unwise to start the Sybase processes if the data was not believed to be suitable for loading.

Two things were required for this automated process:
1. A data/process check procedure that could identify if there were any critical problems in the batch. If any were found, the flag file would be created with a 'no go' indicator set. Otherwise a 'go' indicator was set and the job scheduler would initiate the build process.
2. The process needed detailed records to be made available to the SAS team. These must include the full batch logs and a series of routine tests of the data quality. The logs would be used to indicate potential problems in the process, and the tests would quickly identify any missing clusters of data in the load.

If the end of the calendar month fell on or after Wednesday in the week, or Tuesday if there was a bank holiday at the end of the week, then the first two phases of the month end process could be finished by the weekend.

If the Sybase processes were run across the weekend, then the exclusive use of the server meant the process ran more quickly. However, it was unreasonable to call in the Sybase DBAs if the data was not suitable for loading.

The 'go/no go' process needed to take the place of a SAS analyst in interpreting the process results. Calling in two staff members over the weekend was wasteful and undesirable, and the DBA could not be expected to read and interpret hundreds of pages of

SAS logs and tables.

CHECKING DATA CLUSTERS
The key questions about the data being loaded revolved around the distribution of the data, and changes in that distribution over time. Testing these 'time based' data clusters involved producing frequency tables that matched either counts or summed values for each class of data against time.

A timeline based on monthly increments matched the data extraction intervals. This was subtle enough to discriminate the timing of the release of new products, or of an increase in claim rates to coincide with natural events such as cyclones or floods. Business changes such as lodgement of new claims, finalisation of existing claims and the purchase or expiry of policies could also be quickly verified with the business areas that understood any recent changes.

The SAS team designed a test program that included these process rules:

1. An error in the core table build processes was fatal. This meant that if the 'Policy' or 'Claim' fact tables were incorrect, then the dimension tables associated with them would also be in doubt.
2. An error in the core table load processes was fatal. If the data for the core tables were not loaded correctly, then DataBase Consistency Checks (DBCC) would fail and demonstrate that the MIS was unreliable.
3. Where data was split between two tables, if the resultant record counts between the two tables did not match the source data, then the data may be unreliable. An error message was written to the log with a data test macro.
4. Where the total financial value of records loaded to the Sybase tables differed from the source data by more than 2%, the Business would refuse to accept the data.
5. Where the total financial value of records differed from previous months by more than the 'reasonable' margin set by the business, then the data needed to be checked.

With more than fifty such tests, it was easy for the SAS analysts to identify the area of concern in the data. However, both the business and the DBAs wanted a simple answer; "Just the facts ma'am".

SUGI in California: regional colour.

In 1952, the television industry presented the first episode in an eight-year series developed from a radio show that began in 1949. The show was revived for another four year run in 1967, with the addition of Harry Morgan, pictured here with Jack Webb (right).

'Dragnet' was based in Los Angeles, and in its brisk 26 minute pace it exposed a crime, followed the investigation and delivered the right answer every time before the end of the show.

Key investigator on the show was Sergeant Joe Friday, whose statement "Just the facts ma'am" may have been a scriptwriters attempt to keep the pace up, or a Directors attempt to keep the show on time.

But it became a well-known saying that might be viewed today as a sign that many people don't care about the details; they just want the solution. How many business people do you know who might have a similar view?

The reports on the process were scaled according to the audience:

1. A 'go/no go' indicator file for the automated process
2. An email to the DBAs reporting either 'data availability' or 'process problem'
3. An email to the SAS analysts reporting the details of any problems found in evaluating the process rules
4. An email to the business team reporting the financial reconciliation and 'top five facts' about the month end process. This email arrived up to five days before they might have had the results to their own reconciliation of the MIS.

To support this functionality, it was also necessary to make the SAS logs and output files available in an easy to access, indexed format. Now, no matter who asked about the month end process, they would get the answer they would expect to the question "Is it safe?"

SUGI in California: a parallel from the film industry.

The film 'Marathon Man' (John Schlesinger 1976) brought together Laurence Olivier and Dustin Hoffman in a gruesome and haunting thriller.

Olivier's cold torture of Hoffman, whose innocence serves only to heighten the brutality of the attack, is a powerful piece of film direction. Olivier demands to know 'is it safe?', while Hoffman has no idea of the meaning of the question.

We might draw a parallel between Hoffman's plight and the demands of business. The questioner needs a one-word answer to what may be a complex issue. The analyst might be asked something they don't fully comprehend.

The information delivery discussed in this paper is intended to show one possible solution. It delivers information in different forms, customised for different audiences. Then each may use personal judgement and experience to know if indeed it is safe.

Since the process at each month end would accumulate 'snapshot' data about the MIS load, it was also possible for issues this month to be compared to previous months to identify the emergence of any problem found. For instance, if new policy numbers appeared to be low this month compared to previous months, then previous months' tables could be compared to establish if there was any retrospective change to monthly data.

With such a wealth of information available, it was likely to remain unused if a method of indexing and quickly accessing key information was not found. The solution was in the creation of web pages. If logs were converted to Web pages, and particular entries such as 'Error or Warning' were highlighted, then scanning the log for an error would not take much time.

Similarly, if a program performed a process a series of times, such as loading each of 30 tables in turn, then scanning down the log to the individual load process required could be easier if the start of the load process was distinctively marked in the log file.

Conversion of list files was also going to be of benefit if the title page were clearly identified with some text attribute (large font, distinctive colour, bold text etc) that was easy to find.

An object-oriented approach to the problem meant simple techniques could be implemented and extended later. The techniques created were:

1. Create an object that tested a data set record count and populated that to a macro token. Using the macro meant comparing the values of a series of macro tokens,

calculating the difference and generating a 'Program Error:' message in the log, if counts did not reconcile. It also meant that where data was split off because of bad values (such as missing dates), the number of bad date records could be used as a flag to create a 'Program Warning:' message.

2. Split the batch into subject areas, with one program dealing with one source and target table.
3. Create a master job that called each program in turn and created a unique log and list file for each.
4. Create a 'test job' that included a macro to scan a log for certain text values such as 'Warning:', 'Error:', 'Program Error:', 'data step was stopped' and accumulate the number of such messages for each log.
5. In the test job, store the error counts to a data set for each log scanned and accumulates an error index for the overall batch.
6. Pass the error index to the object which created and populated the 'go/no go' file.
7. Create an object to write an email message. Pass it a parameter for the address and it will write two files, one (a command file) containing a Unix command to send an email message to an address with a given text file as message content, the second being the message content file.
8. Create an object to submit the Unix command file, pause and then check the email system log for satisfactory completion of the email process. Then write a message to the SAS log reporting the message number and transmission date/time.
9. Create an object to parse text strings for certain values, and then embed HTML formatting commands within the strings.

Most objects were SAS macros. The principal benefit of the 'object-oriented' approach was seen in the file parser object.

THE FILE PARSERS.

The first version of the log parser simply sought particular strings, such as the 'Error:' text appearing in the log, and added HTML tags to change the colour and appearance attributes.

However, as the later tests of the batch process turned up more issues, such as space failures, Sybase database unavailability, stack overflows or operating system errors, many appearance attributes needed to be adjusted.

The code for the log parser looked like this:

```
Data WHATLOG;
  Length PUTSTR $132;
  Infile 'a6.log'  Truncover  End = LAST;
  File 'a6.html';
  Input @1 READSTR $Char132.;
  If _N_ = 1 Then Do;
    PUT "<HTML><HEAD><FONT SIZE=6 COLOR=BLUE>
         <DIV ALIGN=CENTER>";
    PUT "SAS generated Log report on A6.";
    PUT "</DIV></FONT><BR><HEAD><BODY>";
    Put "<HR SIZE=3 WIDTH=100% NOSHADE
         ALIGN=CENTER COLOR=GREEN>";
  End;
  If READSTR Eq: 'NOTE:' Then PUTSTR =
     '<STRONG>' || Trim( READSTR) ||
     '</STRONG><BR>';
  Else If READSTR Eq: 'ERROR' Then PUTSTR =
     '<FONT COLOR=RED>' || Trim( READSTR) ||
     '</FONT><BR>';
  Else If READSTR Eq: 'WARNING' Then PUTSTR =
     '<FONT COLOR=ORANGE>' || Trim( READSTR)
     || '</FONT><BR>';
  Else  PUTSTR = Trim( READSTR) || '<BR>';
  Put PUTSTR $Char132.;
  If LAST Then Do;
    Put '<HR><TABLE WIDTH="100%"><TR>';
    Put '<TD WIDTH = "50%"><ADDRESS>';
    Put "<DIV><FONT SIZE=2>Developed in the
         SAS System Viewer editor by<BR>";
    Put "David H. Johnson, Business
         Information Systems Consultant
         <BR>";
    Put '<A HREF="Mailto:
         sasuser@dkvj-cons.com">DKV-J
         Consultancies Ltd</A>';
    Put "@ Client site, July-Nov 1999.<BR>";
    Put "+44 (0)7080 81 8399<BR>";
    Put 'C/- "Bonds Cottage", Holmeswood Rd,
         Holmeswood Nr Rufford,';
    Put "Lancashire L40 1UA UK<BR></ADDRESS>
         </TD>";

    Put '<TD WIDTH = "50%" ALIGN = "RIGHT">
         <BLOCKQUOTE>';
    Put '<IMG SRC="ribbon.gif"
         ALIGN="RIGHT">';
    Put "<STRONG><FONT SIZE = 3>... Built in
         SAS<BR>";
    Put "</FONT></STRONG></BLOCKQUOTE></FONT>
         </DIV></TD></TR></TABLE>";
    Put "<!-- SAS code definition, David
         Johnson  DKV-J Consultancies
         29 September 1999 -->";
    PUT "</BODY></HTML>";
  End;
Run;
```

The early appearance designed into the logs was also 'cheap and cheerful', since the program creators were inexperienced Web designers at that time. They also did not have the time to refine this appearance given the delay it would cause to the completion of the writing of the batch code.

The solution was found in Cascading Style Sheets, and the log parser was changed to add CSS name attributes to the marked up text. Each name reflected the type of condition it was marking such as 'SasLog.Error' and it was then the responsibility of a more experienced web page designer to refine the Style Sheet and produce web pages that were easier to work with.

The log parser was also analysed and found to contain five key parts:

1. File naming and web page header creation
2. Log specific text formatting
3. Navigation bar creation
4. Page footer appearance with navigation bar and page 'signatures'.
5. Recording the page name and address in a data set.

These were split off into their five component objects, and then parsing the list or program file required only the definition of a text-formatting program specific to the expected content of the list or program file. Figure 2 represents the HTML creation process.

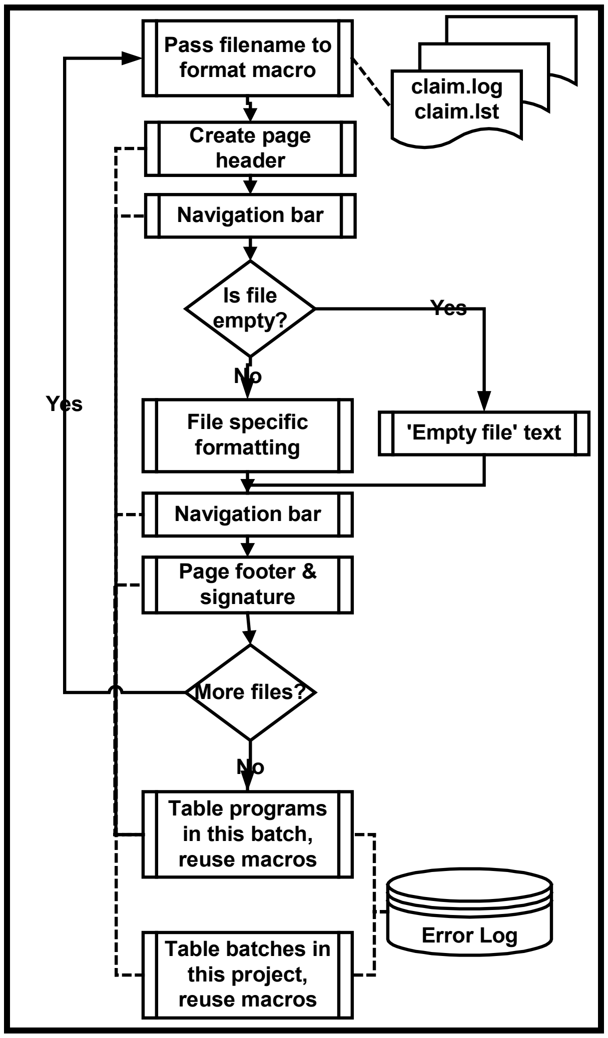

Figure 2

As a batch completed, the name of each program that had been run was passed to the file parser master program. The presence of each expected file was established with a 'file existence test' object. If a file was missing, such as an empty list file, then the default web page creation object was called. This created a page with the text message 'this file is empty'. Otherwise the five parser objects were called in turn and web pages written to the web page destination.

Once all files had been processed, a 'batch index' page was created. It contained the batch start and end times, and a table reporting the start time for each component program. Against each time was a hyperlink for each of the program, log and list files. Where a log contained an error or warning, the table cell was highlighted to identify it clearly.

Finally, a 'project index' page was created. This contained the date/time and run time of each batch that had been run for the MIS project, and a hyperlink to the 'batch index' page. Where the batch had any errors or warnings, that cell in the table was highlighted. The web pages were now of use in this manner:

1. The business could review the run dates for the latest months of the MIS build, see the expected delivery time and identify to what extent it was increasing over time
2. The Project Manager could get a similar perspective and see how often the process had been run with problems
3. The SAS team could see from the top page whether the batch run was 'clean'
4. They could 'drill down' to each batch and see which parts of the batch had problems and whether the run time for any batch component was changing markedly over time
5. They could also drill down to the component levels of the batches and review the log or list for any individual component. Opening the previous month's equivalent file only involved opening a second web browser and clicking once on each of the index pages presented.

WOULD THE OUT2HTML MACRO HAVE DONE AS MUCH?

The decision to create the custom HTML format objects was not taken lightly. The time involved is not trivial, and the benefits needed to outweigh the delays it introduced to completion of other parts of the project. Clearly, the standard macros did not customise the formatting to the same extent, and there was still a need to build the indexing and index page creation objects.

However, the time to drill down to a problem and navigate a log file that may be 200 pages long showed substantial timesaving over more traditional methods of reviewing logs. For response to month end problems, when timing was critical, this saving of a SAS specialist's time to the project was achieved every month.

It also produced other benefits; such as the occasion when a business user wanted information on product levels by month over the previous two years. The MIS was still two days short of release, but the data was provided from the data test summary tables in a matter of minutes.

The object-oriented approach also demonstrated its value when a second data mart was being developed. The same parser objects were reused in the new project with only minor changes to generalise their use. This data mart was run on a weekly basis, and speedy diagnosis of problems was even more critical. Two further data marts were expected in the subsequent 12 months, and similar savings were expected to those projects.

TESTING VERSION 8

The Output Delivery System (ODS) possibilities of the SAS System Version 8 were a clear contender for the output process. However, the project began very shortly before the Year 2000 change freeze began, and a version change was out of the question at that time.

Version 8 was finally tested in the late stages of the project, but the benefits of ODS were not as clear cut and still required some of the other formatting techniques that had been developed. Had it been available in the development phase of the MIS build programs, the clear indexing and simple interface would have added some benefit during the code design, but the production run required other techniques.

One interesting outcome was that version 8 finally provided the FileName Email engine for the Unix platform that had been available for Windows since Version 6.10. However, while the method of sending email was much simpler, it did not include any clear diagnostic processes. The 'email send' object (created for the project) tested the Unix system files for email completion. It was a much more robust solution and proved to be more desirable in the long term.

CONCLUSION

The information delivery for the project was in a cascaded model. It used email and HTML pages generated by the SAS System to enhance the value of the project's front end. The levels of information were:

- Drillable HTML page index of all updates
- Drillable HTML page index of the elements of each batch run
- HTML page describing each MIS build program

- Analysis of the data in each core area
- The program used for delivery of each core area
- Email reporting batch start to support staff
- Email reporting batch completion & success to support staff
- Email reporting reconciliation to support staff.
- Email reporting summary reconciliation to the business
- Email releasing (or holding) SAS built data for the MIS load

All this was done with a handful of reusable SAS macros that were reused with great success in other MIS projects.

TRADEMARK CITATION

REFERENCES

SAS Institute Inc (1995), *TS460: Accessing External DLLs with SAS 6.1x for Win32s*, Cary, NC, SAS Institute Inc.

Barron, David L & Hemedinger, Chris (1996), "Accessing Dynamic Link Library Routines with the SAS System for Windows", *Observations: The Technical Journal for SAS Software Users*, First Quarter 1996.

SAS Institute Inc (1996), *Microsoft Windows Environment: Changes and Enhancements to the SAS System, Release 6.11*, Cary, NC, SAS Institute Inc.

Johnson, David H (1997), *DLLs, APIs and SAS*, SAS Melbourne User Group.

SAS Institute Inc (2000 et al), "SAS Companion for the Microsoft Windows environment", *SAS On Line documentation*, Cary, NC, SAS Institute Inc.

Johnson, David H (2000), "Have SAS, will travel", *SAS European User Group International*, Dublin, SAS Institute Inc.

ACKNOWLEDGEMENTS

My clients who have asked for the robustness and functionality that required this development to take place.

The SUGI committee and especially the Information Visualisation section chair, Carnetta Francis-Minor, for supporting the production of this paper.

The SAS Institute Inc for their documentation, and the 'Observations' publications which provide a forum for sharing ideas and experiences.

The helpful people who post ideas and assistance to SAS/L, the global SAS discussion list you can access at http://www.listserv.uga.edu.

Last but not least, my family for their support and patience.

CONTACT INFORMATION

Your comments, suggestions and questions are valued and encouraged. Please contact the author:

David Johnson
DKV-J Consultancies
C/- 'Bonds Cottage,
Holmeswood Rd
Holmeswood nr Rufford
Lancashire England L40 1UA

Work Phone:	+44 (0)7080 81 8399
Fax:	+44 (0)7092 25 9556
Email:	sugi168@dkvj-cons.com
Web:	http://www.dkvj-cons.com