Paper 152-27

From Manual to Automatic with Overdrive - Using SAS® to Automate Report Generation

Faron Kincheloe, Baylor University, Waco, TX

ABSTRACT

This paper is a case study of how SAS® products were used to automate the creation of a report. The original version of the report was created in Excel. Data were taken from three separate SAS® programs and entered by hand into the spreadsheet. The layout was not a standard format that could be readily replicated from a SAS® procedure. The conversion process involved the creation of a data mart using SAS® Data Warehouse Administrator. Warehouse Administrator was also used to summarize the data into a summary dataset. The SAS® Output Delivery System was then used to create an HTML document containing the desired information. Finally, the data warehouse scheduler was used in conjunction with Windows NT to automate the loading, summarization, and output processes. A new report is created every Sunday morning and placed on the university intranet without any human intervention.

An interactive version is available for the user to create a customized version of the report. This version uses SAS® Intrnet to pass parameters from a web page to the SAS® program that generates the output. The output is then returned to the user's browser in HTML format.

INTRODUCTION

The Office of Information Management and Testing Services (IMTS) at Baylor University is responsible for the creation of a number of reports to support University decision- making. One of the areas supported is the recruiting and admitting of new students. This paper discusses the process IMTS used to automate a report that was previously produced manually. The report was a comparison of various admissions statistics from the current recruiting period and the previous two years. It was created every two weeks using Excel and was distributed through campus mail. (See Exhibit 1 for a sample of the old report.) The result of the conversion was to be a report in HTML format that could be accessed through the University data warehouse. (See Exhibit 2 for sample output.) The report recipient would be able to go directly to the statistics from the current week or dynamically create a custom report to compare as many as three separate weeks from the past.

The data were stored in SAS® datasets on an IBM mainframe. Each dataset represented a snapshot of applicant data taken at the end of a particular week. In order to create the report in Excel, a reporting program was run one time for each of three datasets. The results of these three jobs were then transferred to the spreadsheet.

THE CONVERSION PROCESS

The first step in the conversion process was to integrate the existing data into the data warehouse. Approximately one hundred twenty datasets had to be consolidated into a single dataset on the Windows NT server where the warehouse resides. A program was written to append the datasets together, one at a time.

DATA PREPARATION

A certain amount of "data cleansing" needed to be done in conjunction with the consolidation. Several fields have been added to the datasets over time. Another field, counselor territory, was not routinely updated in the past. The value in this field is derived from the zip code of the applicant's permanent address. Until recently, there was no procedure in the transactional system to recalculate the territory when the address was changed or a new address was received. Therefore, there was no confidence in the accuracy of the territory value in the frozen datasets. Fortunately, the zip code was available to use in computing the territory value in the program that consolidated the datasets. The datasets were added, one at a time, according to their chronological age, starting with the newest first. This was the simplest way to determine which fields were missing from the older datasets. When the program encountered a field that was missing in the original dataset, it would halt. The job log would contain a listing of the field or fields that caused the program to stop. In most cases, the program could be modified to derive the values for the missing fields from the existing data. Otherwise, the records were added to warehouse with a null value for the missing field.

Some additional fields were created in the consolidation process. Most of these were flags to facilitate reporting and analysis. In some cases, the analysis tools do not do a good job of counting occurrences of different values in a text field. To get around this problem, a "flag" was created. The flag is a numeric variable. It is populated with the value of one (1) whenever the text value to be counted occurs. An example of this is the Accepted Flag. The value is set to one (1) if the applicant has been accepted. Otherwise, the value is set to null. In order to determine the number of accepted applicants, we simply have to find the sum of values in the Accepted Flag field.

Another field that was needed was a value to identify from which dataset each record originally came. Now that all records from all snapshots were included in the same dataset, there needed to be some way to distinguish among the records. An applicant could potentially appear in the dataset up to 52 times. A Week field was created to identify the groups of records. This field is actually a compound of the week and year in which the snapshot was taken. For example, records from the third week in the Fall 2000 recruiting year would all have the value of 2000-03. As new records are appended to the warehouse, they are also assigned a Week value to identify the time period in which the records were added. Once all of the weekly datasets were consolidated into one and the required fields were added, the new dataset could be used as the source for the initial load of data into the data warehouse.

ROLE OF THE DATA WAREHOUSE

The data warehouse is managed with a software package called SAS® Data Warehouse Administrator. The Data Warehouse Administrator is graphical interface that enables warehouse manager to build a warehouse in a "point-and-click" manner. It provides a mechanism to manage the metadata that documents the warehouse. It also provides a graphical representation of the structure of the warehouse. (See Exhibit 3.) The Data Warehouse Administrator can even generate code to populate the various tables in the warehouse. Data Warehouse Administrator is a component of the SAS® software suite and uses other elements of SAS® software to perform its various functions. The SAS® ACCESS module is very versatile in its ability to communicate with various types of databases on various platforms. This makes the SAS® software suite an ideal tool to extract data from the transactional system on the mainframe and store it in the warehouse on a Windows NT server.

The consolidated dataset that was moved into the data warehouse contained every field that contained relevant data about an applicant. Many of these fields were not required to create the report that was being converted. However, the fields were retained in the warehouse in case they were needed for analysis and reporting at a later date. A detail table was created

that was a subset of the first dataset. The detail table was stored in a separate dataset and contained only the fields that were needed for the converted report. The creation of the detail table is a straightforward mapping of fields from one dataset to another. Therefore, the Warehouse Administrator software was allowed to automatically generate the code to process this table.

With all of the data in place, the main step that remained was to produce the desired output in the proper format. The code to produce the output was first written in the SAS® programming interface. At this point, all output is directed to the output window of the interactive SAS® session. Once the program is completed and debugged, some minor modifications can be made so that the code will run in the appropriate environment. In the completed project, there are actually two versions of the program. One version is designed to accept certain parameters from a web form and then return the output back to the browser. This program runs within the context of SAS® Intrnet and allows the user to create customized report from historical data. The other program is executed by the Windows scheduler to refresh the report showing the current week's statistics. Its output is written to a static HTML file that can be accessed over the university's intranet.

There are several procedures in the SAS® language that produce output in tabular format. Most of the calculations could be performed using the Tabulate Procedure. However, when the calculations were made using these procedures, the layout was not like the layout in the Excel report and was not conducive for making comparisons. The procedures in SAS® are primarily geared toward reports where the columns are totaled and percentages are computed based on the column totals. This report did not fit into that pattern.

It was determined that the best way to get the desired results was to create a summary table and then create the report from the summary table. There are four different types of applicants in each dataset; fall freshmen, spring freshmen, fall transfers and spring transfers. Each type is identified with a distinct code such as FR for fall freshmen. All of the summary computations needed to be grouped together according to applicant type and week. A "key" field was created by concatenating the applicant type field with the week field. Even though the key was not a unique field in the detail data, the summary table would have only one record per key value.

SAS® provides a feature that allows SQL functions to be used within a SAS® program. This feature was used to compute the summaries. Several views were created, one for each analysis element. For example, one view would compute the number of males in each stage of the application process. The value in the new field, APGMNUM, was computed by finding the sum of the applicant flag field where gender is male, grouped by the key field as shown in the sample code below:

```
proc sql; *calculate number of males at each
stage;
create view work.vmale as
select pkey, sum(appflag) as APGMNUM,
sum(acptflag) as acgmnum, sum(depflag) as
dpgmnum,sum(confflag) as cfgmnum
from keyed
where sex='M'
group by pkey;
quit;
```

All of the views were then joined on the key field into one big table. Percentage fields were created in the merging process. For example, the percentage of male applicants was computed by dividing the value in APGMNUM with the total number of applicants for each key value. Similar calculations were done for ethnicity, religion, and state of residence. The Warehouse Administrator also handles the code for loading this summary table. However, since it is custom code, the process type selected was "User Written" instead of "SAS Generated." This allows our program code to be inserted into the data loading sequence.

One feature of the report that the users desired was the ability to compare actual results with goals in certain areas. For instance, the Admissions Department may have a goal to attract a certain percentage of out-of-state students. They wanted to see the actual ratio of out-of-state applicants compared to the goal. The goals are set once a year and remain the same the entire year. A "goal" dataset was created manually. This dataset had the same fields as the summary dataset. The "key" value was created by concatenating the applicant type with the word GOAL. Where a goal had been set the number was keyed into the dataset. Each time the summary dataset is refreshed to include data from a new week, the goal dataset is appended to the bottom of the summary dataset. When the report is created, the "where" clause specifies key values that include goal and the three week values being compared. The sorting works out so that the goal value is on the top and then the week values in descending order from most recent to oldest.

The original report included a summary section that was created on a word processor. The essence of the summary was a comparison of the current year with the previous year. The percentage change was manually calculated for each parameter being compared. The summary was not to be included in the new report format, but the users still wanted to see the comparison data. The summary was replaced with a table to accommodate their request. The automation of these calculations was difficult because we were comparing values from two consecutive rows in the dataset. The transpose procedure was used to convert rows to columns and columns to rows:

```
proc transpose data=work.frapps out= FRCHANGE
name=Stage prefix=&appltype;
id week;
var applynum acptnum depnum confnum;
```

The result of the code above is a dataset named FRCHANGE. Percentage calculations are made in a subsequent data step using FRCHANGE as the source dataset. The print procedure is then used to create the comparison table in the report. As an alternative, the report procedure could be run against the FRCHANGE dataset to produce similar results.

AUTOMATION

The final step was to automate all of the processing. The SAS® Data Warehouse Administrator software has a scheduling module for this purpose. The scheduling module is actually just a graphical interface that passes commands to the scheduling utility of either a Windows NT or Unix server. The update and summarization jobs were scheduled to run in the early hours of each Sunday morning.

Version 1.3 of the Data Warehouse Administrator, which ships with SAS® version 6, is still being used at Baylor. As a result, all jobs that are scheduled through the Data Warehouse Administrator are executed using SAS® version 6. In order to get the report to look the way we wanted in a web page, we needed the power of the Output Delivery System (ODS) in SAS® version 8. The HTML output macros that are available for version 6 offer little flexibility in the way the output is presented. The final job that actually creates the report output had to be manually entered into the NT scheduler and configured to run under SAS® version 8. By 6:30 each Sunday morning, the warehouse update jobs have completed and the web page is overwritten with a new report containing the latest admissions statistics.

CONCLUSION

The automation of this report created a number of benefits for both its creator and its recipients. The manual report was prepared on Monday of every other week. Average preparation time was three to four hours. Paper copies were distributed through campus mail and usually reached the recipient by

Tuesday afternoon. Automation of the report gave the author six to eight hours a month that could be spent on more productive activities. Since data is no longer moved by hand from one report to another, the chance for human error has been eliminated. The report is more up to date because it is generated weekly instead of every two weeks. It is also available to the recipient almost two days earlier. Paper copies are easily lost amid the clutter of our desks. The automated version is always in the same location and is as easy to find as a bookmark on the web browser.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

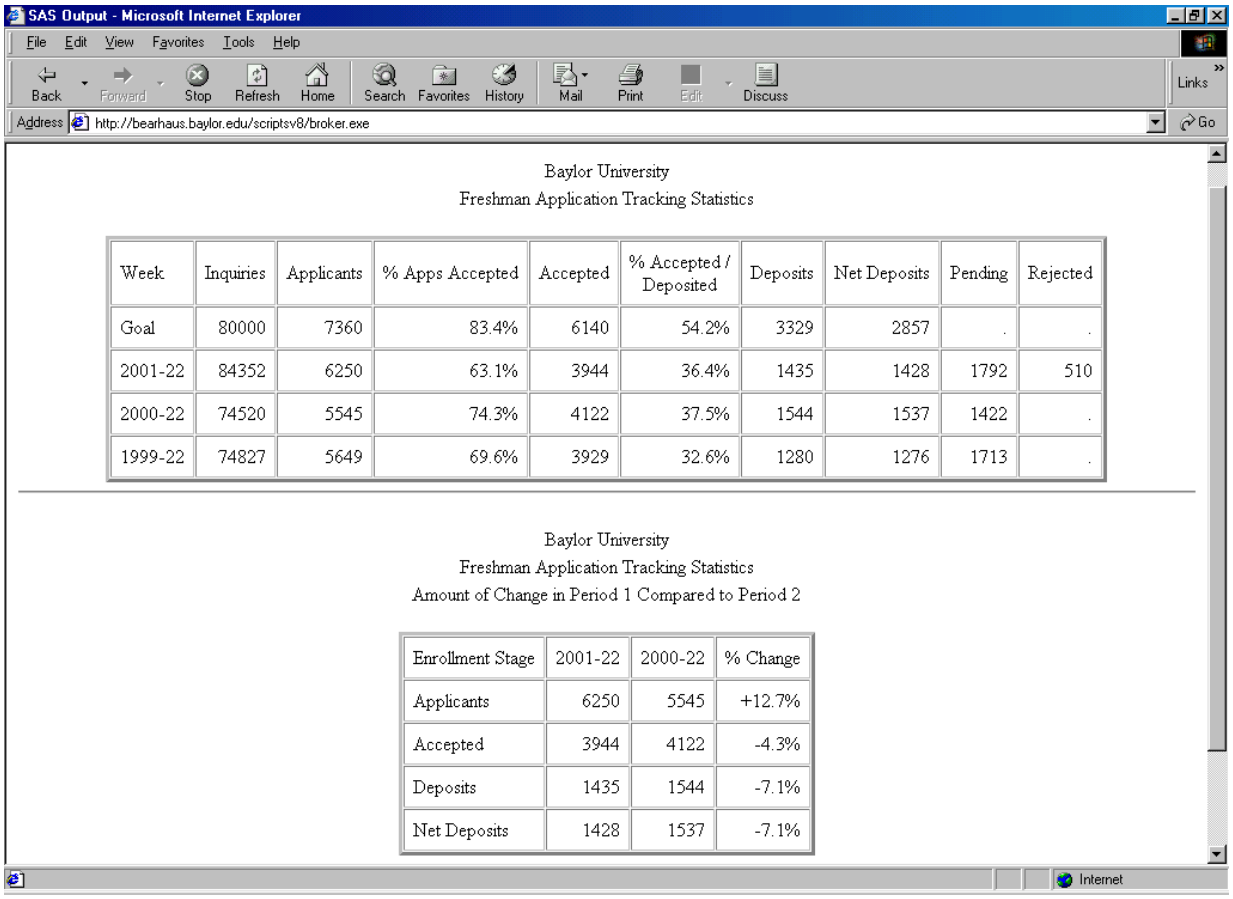
Faron Kincheloe
Baylor University
P.O. Box 97032
Waco, TX 76798
Phone: (254) 710-8835
Email: Faron_Kincheloe@baylor.edu

Exhibit 1 – Sample from Old Excel Report

Baylor University
Summer/Fall New Freshmen Application Tracking
End of Week 22 Statistics - February 2, 2001

	Inquiries	Applicants	Accepted	Deposits	Net Deposits	Enrolled in Fall
Summer/Fall 2001 Goals	**80,000**	**7,360**	**6,140**	**3,329**	**2,857**	**2,800**
Summer/Fall 2001	84,352	6,250	3,944	1,435	1,428	
Summer/Fall 2000	74,520	5,545	4,122	1,544	1,537	2,832
Summer/Fall 1999	74,827	5,649	3,929	1,280	1,276	2,772
By Gender						
Male						
Summer/Fall 2001		2,381	1,455	503	502	
Summer/Fall 2000		2,084	1,500	572	569	1,163
Summer/Fall 1999		2,152	1,443	432	431	1,099
Female						
Summer/Fall 2001		3,869	2,489	932	926	
Summer/Fall 2000		3,461	2,622	972	968	1,669
Summer/Fall 1999		3,497	2,486	848	845	1,673

Exhibit 2 – Sample from New Online Report



Baylor University
Freshman Application Tracking Statistics

Week	Inquiries	Applicants	% Apps Accepted	Accepted	% Accepted / Deposited	Deposits	Net Deposits	Pending	Rejected
Goal	80000	7360	83.4%	6140	54.2%	3329	2857	.	.
2001-22	84352	6250	63.1%	3944	36.4%	1435	1428	1792	510
2000-22	74520	5545	74.3%	4122	37.5%	1544	1537	1422	.
1999-22	74827	5649	69.6%	3929	32.6%	1280	1276	1713	.

Baylor University
Freshman Application Tracking Statistics
Amount of Change in Period 1 Compared to Period 2

Enrollment Stage	2001-22	2000-22	% Change
Applicants	6250	5545	+12.7%
Accepted	3944	4122	-4.3%
Deposits	1435	1544	-7.1%
Net Deposits	1428	1537	-7.1%

Exhibit 3 – Data Warehouse Administrator Screen