Paper 132-25

Building a Low Cost SAS® Data Warehouse on NT Workstation

Zhuan (John) Xu, 1ST Consulting LLC, West Des Moines, Iowa

ABSTRACT

More and more companies now understand the benefit of data warehousing. However, a typical warehouse project uses expensive hardware and software. And it usually requires a team of developers. As the PC hardware becomes much more powerful and cheaper, and as the NT operating system becomes more stable, it is now feasible to develop a small scale data warehouse project on PC platform. Since SAS/Base provides the much functionality needed for a warehouse project it is a good fit for such project.

This paper presents a case study of the Medicare Data Warehouse at WellMark Blue Cross Blue Shield of Iowa. It contains the database, a front end for data extraction, a tool for data analysis, and an application designed to solve a special business issue. This paper will discuss a few issues related to such project and some techniques used.

INTRODUCTION

The Medicare division of the WellMark Blue Cross Blue Shield of Iowa handles Medicare Part A claims for many states. In addition to the claim transactions, it has Medicare Review, Anti-Fraud, and other teams to safe guard the system. Such work requires extensive data analysis. The annual data volume is about 7 million claims with about 40 million line items. Previously, the analysis was performed on the same IBM mainframe system running the transaction. The work was often resource demanding and time consuming. And getting data from mainframe to PC for further analysis was difficult due to slow network connection to the mainframe. Obviously, a data warehouse is a very useful tool to increase the productivity.

DESIGN OF THE DATA WAREHOUSE

The data warehouse is designed to meet the following requirements.

- Meeting business needs of data access, data management, and analysis.
- Low cost.
- User friendly user interface.
- Easy to maintain and support.

Although other alternatives, such as UNIX workstation was considered. PC platform was selected based on the above requirements. The hardware for the warehouse was a 400 MHz Pentium II PC running NT workstation with SAS/BASE and SAS/STAT. It has 3 hard disks with about 27 gigs of disk space and 128 Megs of RAM. This was the top of the line in early 1998.

As SAS® has been used on site successfully, no other software was considered. SAS® is used to host the database and for programming and analysis. The user interface was developed with Borland Delphi. Delphi is a PC application development tool well suited for user interface development. Extensive review was conducted to identify the best file on the mainframe to use as data source. The final decision was to use the most complete history data. This way we avoided the need for developing another mainframe data selection program. It also gave us the option to add any data element needed without going back to the mainframe.

In the initial design, Star-Schema Model was used. This approach was selected for its simplicity. However, due to large size of the factor table, we soon realized we have to normalize it into claim table and detail table. MDDB model was considered but was not used, as the business needs required the detail data.

The data warehouse has 4 parts:

- A procedure for loading data from mainframe to the SAS® databases on the NT workstation.
- A front end for data extraction
- A tool for general data management and analysis
- A special application for solving a more complicated business problem

IMPLEMENTATION

The Medicare Data Warehouse was developed in about a year. The author worked part time as a consultant and did most of the work. We first built the data warehouse. Then added a front end for data extraction. Finally, we revised the tool for data management and analysis, and also ported the special application.

The following describe more details of the implementation.

LOADING DATA INTO WAREHOUSE

Creating the data warehouse is the most complicated process. The company already has cartridge reader on an old HP UNIX system; it was thus used to loading the data. In the original design, the data on the mainframe cartridges was to be transferred to 4-mm tapes through another UNIX computer with a cartridge reader. We would then load it to the older HP UNIX within department. The data then should be transferred to the PC through local area network. However, we encountered an unexpected mechanical problem and soon it became a headache. Due to heavy workload reading several dozens of compressed cartridges, both internal and external tape reader on the older HP UNIX system failed. Another external one was added but failed soon. After some research and testing, the data loading process was revised to bypass the old HP UNIX workstation completely. The cartridges were read and saved to the

main UNIX workstation after piped through the UNIX compress command. This reduces files to 10% of their original size. The files in compressed format were then transferred using FTP utility to the PC through a wide area net work (WAN).

It is worth to mention that HP UNIX could not convert mainframe variable length EBCDIC file into ASCII format correctly. As a result, the raw data was kept in the binary format. On the NT workstation the raw data was uncompressed and converted into SAS® datasets with proper infile options and special informats.

Although much research was conducted on the data quality not much data cleansing and transformation was performed during the loading. The business need required the data as is since an invalid entry may represent an issue to be dealt with. Instead, the data-cleansing rule was enforced in the data extraction. The user can select which rule to apply.

This did produce a problem during the initial testing. Users created some simple reports, but some of the reports were truncated. Yet the detail data suggested the report should contain more listing and the report file size suggested that the file was actually bigger than displayed on the screen. It is soon determined that some character variables contain certain invalid characters. These characters might be harmless on the mainframe. But on the PC with ASCII system, some were control character that made anything after it invisible. To fix such problem and to avoid similar problems, all character variables were checked and all invalid characters were deleted during loading.

FRONT END FOR DATA EXTRACTION

To provide data access to the users with different skill set, a three-level data access is designed for the medical staff, the analysts, and the programmers.

A Windows application – Easy Extractor, was developed for users to extract data. It has one main window with 4 tabs. The first tab (Variables) lists variables in claim files and in detail files. The second one (Selection) allows users to create commonly used selection criteria. The third tab (Ad Hoc) is for more complicated selection criteria. It uses a built in where clause builder to produce selection not covered on the selection tab. It was designed for analysts who have experience dealing with MS Access or similar tools. The last one (Note) shows the generated SAS® data extraction program. The program can be used for documentation purpose. It also allows experienced users to modify and submit the generated SAS® program. This way, a SAS® programmer can use the resulting SAS® program as a template to create more complicated selection criteria or to add additional code to perform analysis. The screen shots of Easy Extractor are listed in the Appendix 1.

The SAS® program for data extraction contains mainly three parts. First two parts are data step views for claim datasets and detail datasets, respectively. The data cleansing rule and selection criteria were applied whenever applicable. The last part is a SQL step that combines the claim level data with detail level data.

A lot work was done to hide the details from the user and to make data extraction user friendly. Another priority was the efficiency. For example, when selecting data based on service data, it is necessary to have 3-month of run out time as it takes time for the providers to submit claims and for the system to process them. This application will add three months to the selected service date interval and automatically pick the data sets needed based on the paid date. As another example, if user selected variables in the claim file but not in the detail file. The resulting SAS® program will skip the second part and thus speed up the process. Additionally, SAS® data step views are utilized to reduce the disk space usage.

TOOLS FOR ANALYSIS -- EASY INTERFACE FOR SAS®

To utilize the data warehouse, simply extracting data is not enough. The users should be empowered with tool to turn data into information. The Easy Interface for SAS® (EI) was evaluated and selected as the tool for analysis. EI's functionality can roughly be classified into three major groups: data management, statistics, and analysis.

Data management functionality includes:

- Copying, moving, and deleting SAS® data sets
- Renaming SAS® data sets
- Browsing SAS® data sets
- Performing SAS® Data Steps with support for subsetting, merging, and other SAS® statements.
- SQL subsetting
- SQL summary
- Sorting data
- Importing data from text files and Microsoft Excel files
- Exporting SAS® data to text files, Microsoft Excel and Word files
- SAS® Format management

Statistics functionality includes:

- Summary
- Frequency
- Correlation
- T-Test
- Paired T-Test
- Analysis of Variance (ANOVA)
- Regression
- Confidence intervals for means and proportion
- Sample size estimation for means and proportion with one and two samples
- Fast statistically valid random sampling

Analysis functionality includes:

- Desktop OLAP
- Benchmark analysis
- Histogram
- Ad Hoc Application

It was also determined during the evaluation that tabulate report is needed. Since I was the developer of EI, it was

fairly easy to add PRO TABULATE functionality to EI. More details of EI can be found in my SUGI24 and SUGI 23 papers.

SPECIAL APPLICATION - MEDICAL POLICY AUDITOR

Although EI provided much general-purpose functionality used with the data warehouse, more complicated business issue required the development of special application. MAuditor is an application to monitor the compliance of medical policies by Medicare providers. The author originally developed for WellMark Medicare Part B on a HP UNIX. This application was revised to work with the data warehouse and adapted to Medicare Part A business. MAuditor exams the claim data to determine the percentage of claims that is deniable by many categories for each medical policy. Its user interface generates calls to the SAS® programs for analysis and the result can be automatically displayed in MS Excel charts and in another data map application.

RESULT

In addition to the mechanical problem mentioned earlier, we also ran into problem with SAS® PROC SQL. SAS® would crash when work with dataset over 2 gig. Fortunately, SAS® technical support provided patch to fix the problem. Overall, the project worked smoothly.

The data warehouse now contains claim data from 1997. It has about 20 millions of insurance claims. The detail files contains a total of about 100 millions of observations.

The Medicare staff performs monthly data loading, plus other maintenance and support work.

FUTURE

Personal computers are becoming more powerful and the operating system is more stable. As we are entering year 2000, we can buy a PC with 1-gig RAM and over 100-gig of hard disk spaces. It can also be equipped with multiple CPU units. The CPU itself is approaching 1-ghz. Such system is several times more powerful than the system used in the case study. Furthermore, faster network, high volume storage device (DVD etc.) are also available at reasonable price. This will make low cost SAS® data warehouse a feasible solution to many companies.

CONCLUSION

This paper presents a case study of a low cost data warehouse. It is demonstrated that, it is feasible to build a functional data warehouse on PC with SAS® system.

Even for large SAS® data warehousing project, a PC data warehouse can be used for proof of concept. It can also be used in the early stage of the development. This is possible as SAS® is available on many platforms and porting a SAS® warehouse from one platform to another platform is relatively easy.

REFERENCES

[1] Zhuan (John) Xu, (1999), Easy Interface for SAS® -- A SAS® Application on the Windows® Platform, *Proceedings of the 24th Annual SAS® Users Group International Conference*, Miami Beach, Florida.

[1] Zhuan (John) Xu, (1998), Using SAS® as an Automation Server in Windows Application Development, *Proceedings of the 23rd Annual SAS® Users Group International Conference*, Nashville, Tennessee.

[3] Peter R. Welbrock, (1998), *Strategic Warehousing Principles Using SAS® Software, Cary*, NC: SAS Institute Inc.

CONTACT INFORMATION

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

Zhuan (John) Xu
400 53rd Place
West Des Moines, IA 50226
Phone: (515) 267-5243
Email: johnxu@home.com

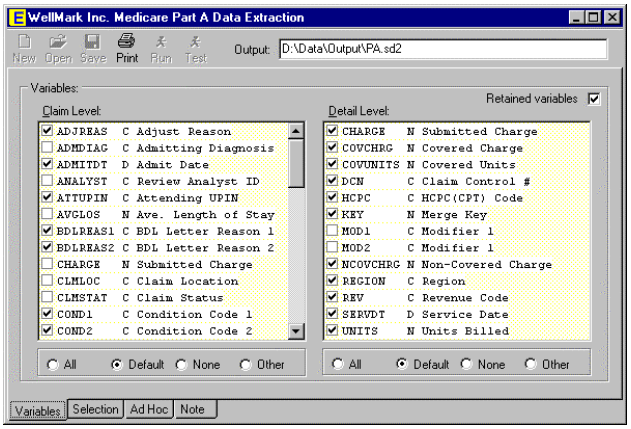
SAS CERTIFIED PROFESSIONAL™ Version 6

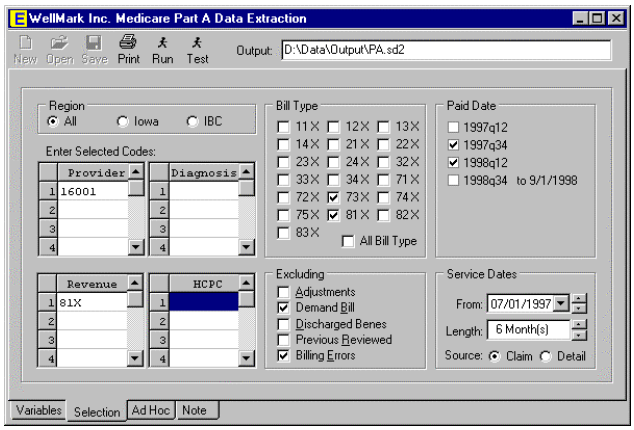
Company info:

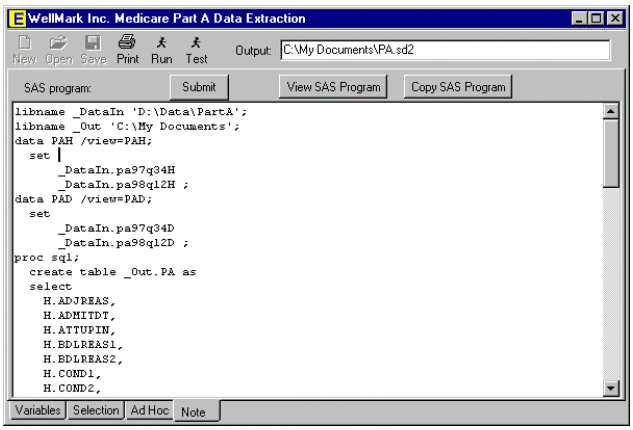
1ST Consulting, L.L.C.
400 53rd Place
West Des Moines, IA 50266
Phone: (515) 778-4093
Fax: (559) 677-8739

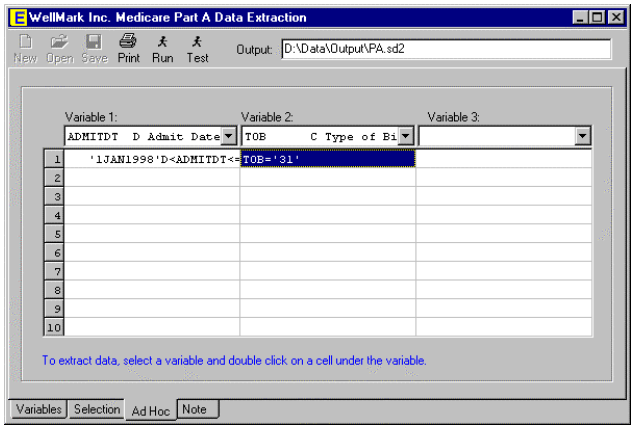
QUALITY PARTNER SAS INSTITUTE INC.

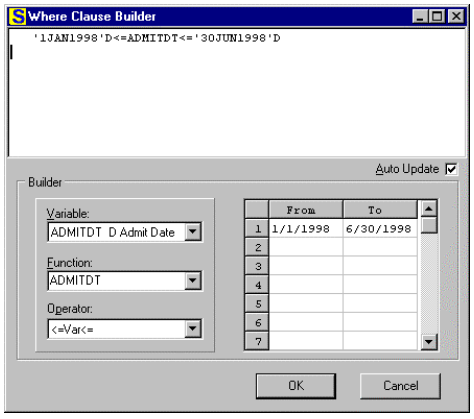
APPEDIX: SCREEN SHOT OF EASY EXTRACTOR




WellMark Inc. Medicare Part A Data Extraction
New Open Save Print Run Test
Output: D:\Data\Output\PA.sd2
Region
All
Iowa
IBC
Enter Selected Codes:
Provider
16001
Diagnosis
Revenue
81X
HCPC
Bill Type
11X 12X 13X
14X 21X 22X
23X 24X 32X
33X 34X 71X
72X 73X 74X
75X 81X 82X
83X
All Bill Type
Paid Date
1997q12
1997q34
1998q12
1998q34 to 9/1/1998
Excluding
Adjustments
Demand Bill
Discharged Benes
Previous Reviewed
Billing Errors
Service Dates
From: 07/01/1997
Length: 6 Month(s)
Source: Claim Detail
Variables
Selection
Ad Hoc
Note


WellMark Inc. Medicare Part A Data Extraction
New Open Save Print Run Test
Output: C:\My Documents\PA.sd2
SAS program:
Submit
View SAS Program
Copy SAS Program
libname _DataIn 'D:\Data\PartA';
libname _Out 'C:\My Documents';
data PAH /view=PAH;
set
_DataIn.pa97q34H
_DataIn.pa98q12H ;
data PAD /view=PAD;
set
_DataIn.pa97q34D
_DataIn.pa98q12D ;
proc sql;
create table _Out.PA as
select
H.ADJREAS,
H.ADMITDT,
H.ATTUPIN,
H.BDLREAS1,
H.BDLREAS2,
H.COND1,
H.COND2,
Variables
Selection
Ad Hoc
Note


WellMark Inc. Medicare Part A Data Extraction
New Open Save Print Run Test
Output: D:\Data\Output\PA.sd2
Variable 1:
Variable 2:
Variable 3:
ADMITDT D Admit Date
TOB C Type of Bi
'1JAN1998'D<ADMITDT<=
TOB='31'
To extract data, select a variable and double click on a cell under the variable.
Variables
Selection
Ad Hoc
Note


Where Clause Builder
'1JAN1998'D<=ADMITDT<='30JUN1998'D
Auto Update
Builder
Variable:
ADMITDT D Admit Date
Function:
ADMITDT
Operator:
<=Var<=
From
To
1/1/1998
6/30/1998
OK
Cancel