



Methods of building an enterprise layer to help data exploitation and delivery

By A. Wilcox
Amadeus Software Limited © 2001



Abstract

This paper will discuss the possible options available to the SAS programmer when developing an enterprise or exploitation layer for a SAS data warehouse. Methods of using SAS Enterprise Guide, SAS Enterprise miner, SAS Enterprise reporter, Base SAS software combined with indexing techniques, SAS/MDDB Server Software and (or) Scalable Performance Data Server, will be discussed highlighting the efficiency drawbacks and benefits to implementing a successful reporting solution.

Introduction

SAS Software is the worlds leading information delivery system. The software offers unrivalled possibilities in gathering and reporting on information stores across many differing applications and hardware platform. This paper will discuss methods of maximising business reporting needs from a technical perspective, to insure that the business builds an effective data warehouse to provide reporting solutions to the business.

There are many software components available within the SAS suite of software products, the SAS software suite offers 'pre-built' reporting solutions such as SAS Enterprise Reporter, SAS Enterprise Guide and SAS Enterprise Miner, Base SAS software also offers programmers the ability to build the reporting solution from the ground up combining the use of Base SAS software and SAS MDDB server software the developer can build a custom suite of very effective reporting solutions.



Understanding The Technology

The 'Pre-Built' Reporting Solutions

The SAS System offers a pre-built suite of applications to help information delivery; these are bundled under the Enterprise Guide Software product, Enterprise Reporter product and the Enterprise Miner product.

Below is information provided by SAS Institute about the function of the three SAS products mentioned above.

SAS Enterprise Guide

Transparent Access to Data

- Access local SAS and non-SAS data types including Microsoft Excel, Microsoft Access
- Access remote SAS data on your SAS Version 8 Servers and other Relational Databases including Oracle and DB2
- Data can be transferred using drag and drop from the SAS server (e.g. transfer a Microsoft Excel file to an MVS data table or transfer a SAS data table from UNIX to the PC)

State of the art Windows look and feel

- Project based
- Provides a point and click interface to the SAS language using easy to use dialogs and wizards
- A new color coded syntax editor is available for more experienced SAS users to modify SAS code
- Full online help

Customisable User Interface

- Fully customizable at user and site level
- Menus and tool bars can be customized to meet the needs of a user

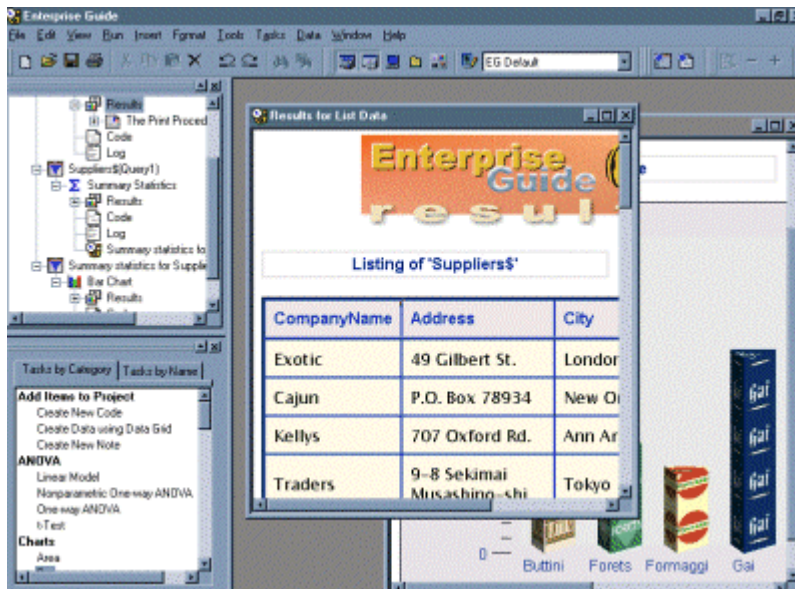
Easy Exporting to Other Applications

- Output can be cut and pasted, dragged and dropped to other applications such as Microsoft Word and Microsoft PowerPoint
- Results may be easily distributed to your e-mail system or saved as HTML and published to the Web
- Results can be printed and/or modified to achieve professional-looking reports and graphics

Automation

- Schedule a project or elements of your project to run at a later time on any of the available SAS System Servers
- Allows for continuous automated publishing of updated data to web sites

Example Screen Shot



(Source: www.sas.com)



SAS Enterprise Reporter

A component of the award-winning SAS Data Warehousing Solution, Enterprise Reporter, completes the process of turning data into useful information by enabling the user to see, analyse and present information customized to their specific reporting needs.

Features Include:

SAS System Integration

Enterprise Reporter software is fully integrated with the SAS System. Therefore, the full power of the SAS System can be utilized to create an end-to-end solution for your information needs.

Ease of Use

Enterprise Reporter software provides an interface designed with a Microsoft Office look and feel. With a familiar palette for report design, users can simply paint, preview and publish reports, while the system takes care of the data.

Simplified Warehouse Reporting

Integration with SAS/Warehouse Administrator software provides an end-to-end warehouse solution by giving the administrator a tool for documenting the contents of the data warehouse. Enterprise Reporter is the users' window to the warehouse fulfilling their information needs for creating reports on paper, or the Web.

Enterprise Wide Report Distribution

Enterprise-wide report distribution is easier than ever before. Various options are available for output such as, HTML for distribution to a web browser, PDF files for distribution via e-mail, or you can simply print a hardcopy of your report.

Reduced Backlog

Designed to give the end user the power to create the reports they need, when they need them, reporting backlogs are minimized, giving IT the freedom work on data administration.

(Source: www.sas.com)

SAS Enterprise Miner

Enterprise Miner helps you reveal trends, explain known outcomes, predict future outcomes and identify factors that can secure a desired effect -- all from collected demographic data and customer buying patterns. The results? Increased revenue, decreased costs and enhanced competitive position.

Enterprise Miner is the first and only data mining solution that addresses the entire data mining process -- all through an intuitive point-and-click graphical user interface (GUI). Combined with SAS data warehousing and OLAP technologies, it creates a synergistic, end-to-end solution that addresses the full spectrum of knowledge discovery.

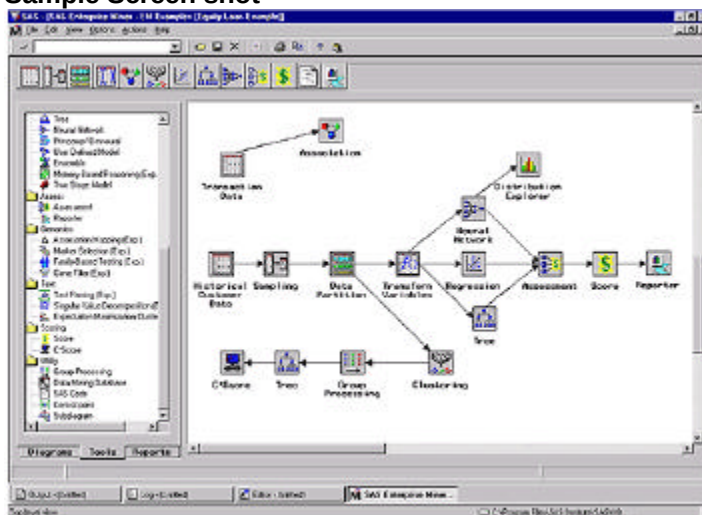
Enterprise Miner delivers on the powerful promise of data mining, enabling you to:

- **Dramatically increase response rates** from direct mail, telephone, E-mail, and Internet delivered campaigns.
- **Identify your most profitable customers** and the underlying reasons for their loyalty.
- **Analyze click stream data** to improve e-commerce strategies.
- **Determine what combination of products your customers are likely to purchase and when.**
- **Detect and deter fraudulent behavior at your e-commerce site.**

And you can manage a wide range of other business challenges, including ways to:

- **Increase customer profitability** through more accurate credit scoring.
- **Apply more profitable ratemaking** in the insurance industry.
- **Save on downtime** by applying predictive maintenance to manufacturing sites.

Sample Screen shot



(Source: www.sas.com)



The Custom Built Reporting Solutions

Base SAS Software

The Base SAS programming language is an extremely powerful data manipulation tool, it allows the programmer to unlimited data manipulation routines using simple instructions sets such as IF THEN processing statements.

There are many benefits of using Base SAS software to analyse data, firstly the code written will be 99% compliant on a range of operating systems such as Windows, Macintosh, OS/2, MVS, CMS, VAX, etc.. In addition the number of records that can be help within a SAS table can reach the terradata scale, the Base SAS programming language is also very intuitive and is used throughout the word in many blue chip companies, making maintenance and code updating relatively routine.

SAS/Access Software

This SAS product allows the SAS software to retrieve information from a whole host of sources. Information stored in Oracle, Ingress, SAP, Excel, Access and many others can now be accessed directly via SAS Software. This product opens the world of retrieving information from a variety of sources.

SAS/MDDB Server Software

This is a relatively new product, which has taken the SAS data-warehousing world by storm, SAS/MDDB Server software offers programmers the ability of producing information 'cubes'.

What is an information cube and why would I want one?

A simple description of an MDDB is that it is very similar to a summarised table built using PROC SUMMARY or PROC MEANS. The major difference between a MDDB and a normal summarised dataset is size and speed, an MDDB is often only 20% -40% the size in physical disc space of the equivalent Proc Summary. It offers unrivalled speed over a standard Sas PROC SUMMARY table when extracting data using a WHERE clause query.



Indexing SAS tables

The simplest way of increasing the performance of a SAS table is to index it, unfortunately there are downsides to indexing.

Firstly the index will require 30-50% of the disk space the actual table occupies, so a 1 GB SAS Table could have an index of 500 MB. The amount of space an index will require varies enormously depending on how complex the index is.

Secondly the index must be rebuilt every time the SAS table is updated making indexing routines strictly part of the overnight or weekly update procedure of the SAS data warehouse.

The major upside is realised when using WHERE clause data extraction routines, minutes in job run-time become seconds.

Pre-Summarising Data

The methods of pre-summarising data using the SAS system are as follows:

Proc summary

This is the defacto standard when producing summarised SAS tables to help form an enterprise layer, the code is simple and straight forward.

Example

```
Proc Summary data=sasuser.sample nway;  
  Class status gender;  
  Var age height  
  Output out=sum.sample sum=;  
Run;
```

The source table is the sample table, the nway options identifies that only the highest level of interaction will be retained. The Class statement lists the levels of summarised interaction, the VAR statement indicates the analysis variables to be summarised and finally the output statement stores the information into a SAS table which will form part of the enterprise layer.



Proc Means

Is basically the same as using a PROC SUMMARY especially in SAS version 7 and there is simply a slight syntax change.

Example

```
Proc Means data=sasuser.sample out=sum.sample sum;  
  Class status gender;  
  Var age height;  
Run;
```

Proc MDDB

This procedure requires the SAS/MDDB Server software product to be licensed. The benefits of using MDDBS are those of speed and physical disk size, again classification and analysis variables are specified as in the other two methods discussed above, only this time a multi-dimensional cube is built. In order to extract information from this 'cube' using Base SAS Software a special libname engine is required.

```
Libnames sasmddb sassfio 'c:\sample';
```

```
Proc mddb data=sasuser.sample out=sasmddb.example;  
  Class gender status;  
  Var age / sum;  
  Var height / sum;  
Run;
```

Now to produce a report simply reference the MDDB cube with the sasmddb libname.

(NOTE: The sasmddb libname is not a reserved word)



SAS SPD Server

Access Data at Breakneck Speed

Storing large volumes of data without hampering system performance is a major challenge for many IT professionals. The Scalable Performance Data Server lets you manage changing enterprise requirements. You can rapidly add more capacity and enhance performance for mission-critical applications -- getting answers 15 to 20 percent faster than before, in most cases.

By utilizing the latest parallel process and data server capabilities, the Scalable Performance Data Server delivers a fully integrated and seamless way to access large volumes of data and serves large numbers of concurrent users. An open data server, it enables extremely efficient data access for hundreds of network clients across multiple processors while giving you:

- **Faster data access.** By reducing the amount of time users wait for results, you get more done in less time -- with the assurance that your systems are running 24/7 with limited down time and business impact.
- **Complete scalability.** Reduce time for storing, organizing and retrieving large volumes of data by exploiting parallel hardware functions. Handle increases in data volumes, changes in your hardware architectures and significant increases in the number of users - effortlessly.
- **Rapid return on your investments.** Exploit your architecture to its fullest extent. You get unsurpassed data storage and retrieval for all your decision-support systems without sacrificing access speed, security, cost-effectiveness or integration with existing environments.
- **Comprehensive security.** Secure data at the disk and network levels for storage or data delivery and enhance your control over access to company-sensitive information - one of your company's most important assets.
- **Easy administration.** IT professionals and business users get tools that are easy to integrate, learn and use. And it's open for easy access to client and Web-based applications from SAS and other manufacturers as well.

The Scalable Performance Data Server combines high-performance data delivery capabilities with a highly scalable solution that secures the integrity of your data, provides cost efficiencies, and leverages your current computing investments -- meeting your organization's needs today and in the future

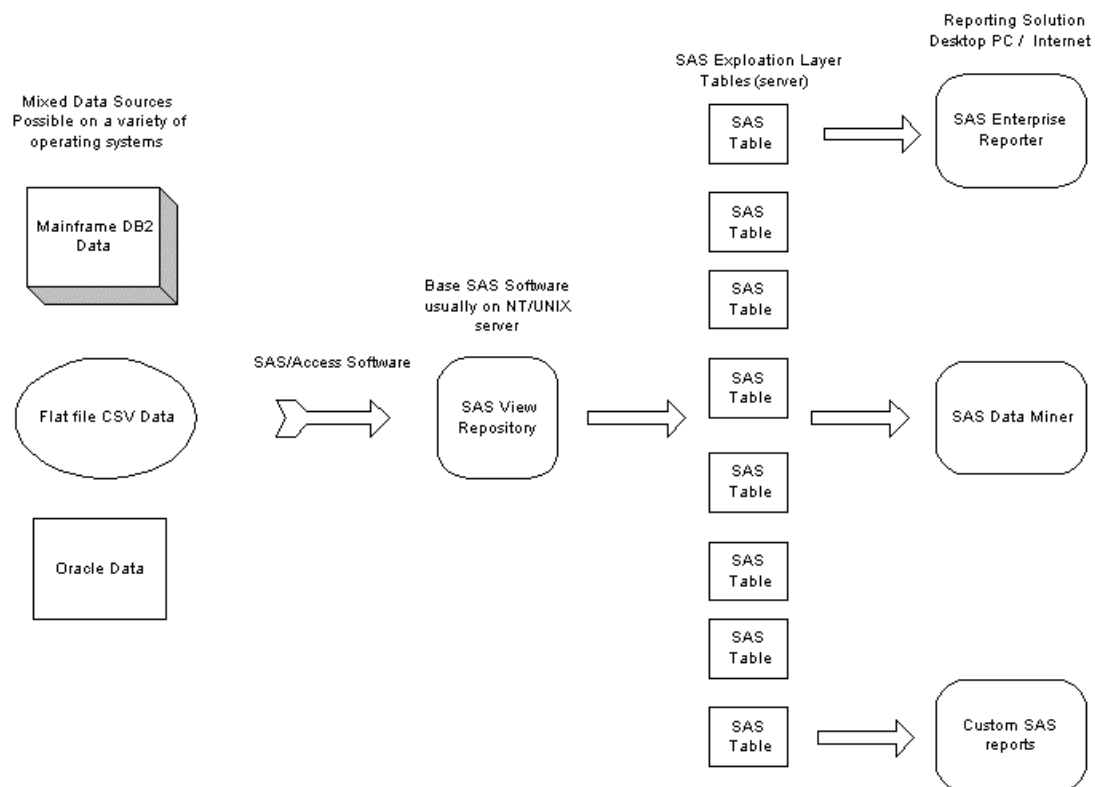
(Source: www.sas.com)

An Enterprise Layer

Firstly what do we mean by an enterprise layer ? Well put simply this is a highly optimized way of storing information within SAS tables to make reporting on the information as efficient as possible.

Businesses hold information in many different data formats. The SAS system sits on top of this information layer to help data exploitation and therefore information delivery. Below is a diagrammatic representation of the structure of a SAS data warehouse and enterprise reporting layer.

Table 1





Building an Enterprise Layer

Step 1 – Building the SAS Repository

The organizational information needs to be accessed by the SAS system. This can be done in two ways either by building SAS tables for the repository (although this means effectively holding the business information twice) or by building SAS views (virtual SAS tables) which will require a minimal amount of space.

Usually the SAS view approach is taken as it is the enterprise layer which will be used to source business queries not the base tables themselves.

There are a number of methods available to the programmer in writing SAS code to build the view, either via Base SAS software, SAS Access or SQL. Below are some coding examples and recommendations as to which methods should be used in specific circumstances.

Example accessing flat file information (i.e. text files)

Using Base SAS Software

Simple Code example

```
Data sasuser.table1 / view=sasuser.table1;  
  Infile 'example.csv' dlm=',' dsd pad missover;  
  Input col1 col2 $ col3;  
Run;
```

The advantage here is the code is simple, every time the sasuser.table1 view is accessed the request for information will go back to the source csv file.

Structured Query Language can also be used to create views, either from existing SAS tables or from other database tables such as ORACLE, DB2, INGRESS etc...

```
PROC SQL;  
  Connect to odbc (dsn=Oracle8 uid=scott pwd=tiger);  
  Create view sasuser.table1  
    as select * from connection to odbc (select * from EXAMPLE);  
  Disconnect from odbc;  
Quit;  
(Provided from SAS web site)
```



One of the main advantages of using SQL to create SAS view is the ability to display the statements used within its definition.

```
Proc sql;  
  Describe sasuser.table1;  
Quit;
```

NOTE

In version 7 you can also use a libname statement with an ODBC engine. Once the libname is assigned successfully, the Oracle table and view can be accessed by using the libref as if it were a SAS data set

```
Libname odbc1 odbc dsn="Oracle7" uid=scott pwd=xxxx;
```

Alternatively Proc Access and Proc Import (v8) can be used to source third party information files.

Examples

```
PROC ACCESS dbms=oracle;  
  Create work.test.access;  
  User="scott"; /* same as connecting from */  
  Orapw="tiger"; /*Sql*Net Client */  
  Path="@V2o8040":  
  Assign=yes;  
  List all;  
  Create work.test.view;  
  Select all;  
Run;  
(Provided from SAS web site)
```

NOTE

In version 7 you no longer need to create access and view descriptors. You can use libname statement with Oracle Engine.

```
libname oralib oracle user=scott password=xxxxxx path=V2o7223;
```



And for a Microsoft Access file format

```
proc import table="customers" out=sasuser.cust dbms=access;  
    uid="thomas";  
    pwd="rocket";  
    database="c:\hrdiv\east.mdb";  
    wgdb="c:\winnt\system32\security.mdb";
```

```
run;
```

(Provided from SAS web site)

To use these procedures you will need to have licensed the relevant SAS/Access product.



Step 2 – Building the SAS Enterprise Layer

This layer is nothing more than a whole host of small SAS tables or MDDB's. These tables are sets of pre-summarized data to help the reporting need of the business.

Once the reporting needs of the company have been understood the SAS developer need to define a whole series of pre-summarized tables, these tables then need to be built/updated. There are a number of methods which can be used to build this pre summarized layer.

Example 1

```
Proc summary data=sasuser.table1(keep=gender status cars age salary) nway;  
  Class gender status cars;  
  Var age salary;  
  Output out=sasuser.layer1 sum=;  
Run;
```

If a higher level of summarization is needed summarize the summary NEVER return back to the base table as this is VERY INEFFICIENT programming, instead follow the example shown below.

```
Proc summary data=sasuser.layer1(keep=gender status age salary) nway;  
  Class gender status;  
  Var age salary;  
  Output out=sasuser.layer2 sum=;  
Run;
```

Example 2

Using the means procedure also offer the programmer summarisation capabilities.

```
Proc means data=sasuser.table1(keep=gender status cars age salary)  
          out=sasuser.layer1;  
  Class gender status cars;  
  Var age salary;  
Run;
```

IMPORTANT NOTE: The use of the CLASS statement within these procedures will cause SAS to perform an internal sort, this means that the table produced will be in sorted order, this is important to note if they are to be used within Match-



Merges as a proc sort might not be required. So on the output dataset use the sortedby option, i.e.

```
Proc summary data=sasuser.layer1(keep=gender status age salary) nway;  
  Class gender status;  
  Var age salary;  
  Output out=sasuser.layer2(sortedby=gender status) sum=;  
Run;
```

In addition the use of the CLASS statement is very memory intensive. If for any reason you do not have enough hardware processing power to complete the step simply change the CLASS statement to a BY this massively reduces memory overhead but will increase the job run-time, i.e.

```
Proc sort    data=sasuser.layer1(keep=gender status age salary)  
            out=work.temp;  
  By gender status;  
Run;
```

```
Proc summary data=work.temp nway;  
  By gender status;  
  Var age salary;  
  Output out=sasuser.layer2(sortedby=gender status) sum=;  
Run;
```

```
Proc delete data=work.temp; *← Tidy up the work area;  
Run;
```



Example 3

The use of the SAS MDDb Server product can reduce the run-time processing of SAS summarisation by as much as 60%, the space required to hold the MDDb data cube is also massively reduced.

```
Proc Mddb  data=sasuser.table1(keep=gender status cars age salary)
           out=example.layer1;
Class gender status cars;
Var age / sum;
Var salary / sum;
Run;
```

```
Libname example sassfio '\\warehouse enterprise layer' access=readonly;
```

```
Proc tabulate data=example.layer1(keep=gender status salary);
  Class gender status;
  Var salary;
  Table status,gender*salary;
Run;
```

The use of the SASSFIO engine within the libnames statement means programmers can access the MDDb cube as if it were a normal SAS Table.

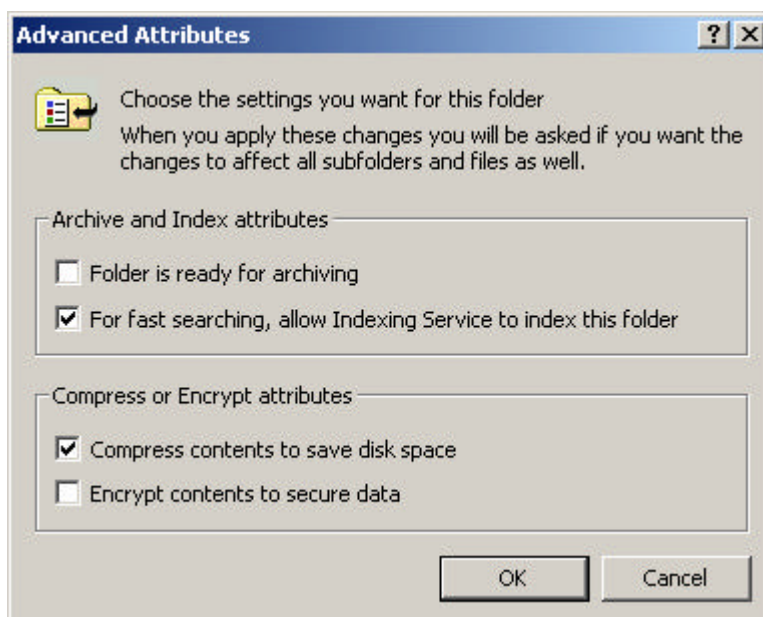
Step 3 – Improving the Performance of the Enterprise Layer

If the enterprise layer has been built using MDDB's then there's not much that can be done to help improve the performance of the enterprise layer. However, if the layer is made up of SAS tables then to attain the best reporting performance INDEX the tables.

SAS indexing dramatically reduces the time needed to process a data request. There are a number of method that can be used to build indexes on your SAS tables, each method differs in run-time depending on the operating system and hardware the SAS programs are running on so its important that all the methods are tested and then the most efficient method employed throughout our enterprise code.

Unfortunately creating an index will often increase the required disk space by 20-50 % depending on the operating system. If adding extra packs is not a viable option then try using the COMPRESS facility the SAS System offers as a data step option. Alternatively, if you are holding your enterprise layer on an NT Server try using the compress option in the properties window of Windows Explorer.

Example



(Microsoft Windows 2000 screen shot, to compress a directory)



Once the space issue has been addressed, deciding on your indexing method is the next hurdle, which programming method should be used to create an index, five methods below are discussed.

Examples

Proc Datasets

```
Proc datasets lib=sasuser mt=data;  
  Modify class;  
  Index create composit=(status);  
Run;quit;
```

Proc SQL

```
Proc sql;  
  Create index status on sasuser.class(status);  
Quit;
```

Data Step

```
Data sasuser.class(index=(status));  
  Set sasuser.class;  
Run;
```

SCL

```
Dsid=open('sasuser.class','V');  
Rc=icreate(dsid,'status',' ','nomiss');  
Rc=close(dsid);
```

Using Proc Append

then

uses proc append to add values to the indexed table.

```
data sample;  
  set sasuser.class(obs=0);  
run;
```

```
proc datasets lib=work;  
  modify sample;  
  index create status;  
  index create gender;  
quit;
```



```
proc append base=sample data=sasuser.class;  
run;
```

Each method has its own merits and all must be tested to help reduce CPU processing times. One immediate improvement in the creation of indices is to make sure the data is pre-sorted by the indexing variables.

Once an index has been defined any where clause syntax applied to the database will take advantage of the index, if the SAS System is using the index a message will appear in the log so long as the option msglevel='I' is active.

Example

A screenshot of a SAS LOG window titled "LOG - (Untitled)". The window displays the following text:

```
4  
5   data Example(index={status});  
6     set sasuser.demog;  
7   run;  
  
NOTE: The data set WORK.EXAMPLE has 104 observations and 15 variables.  
NOTE: The DATA statement used 1.27 seconds.  
  
8  
9   options msglevel='i';  
10  
11  proc print data=example;  
12    where status='S';  
13  run;  
  
NOTE: Index STATUS selected for WHERE clause optimization.  
NOTE: The PROCEDURE PRINT used 0.0 seconds.
```



Step 4 – Using Scalable Data Performance Server

For very large SAS tables, many millions of records in size, the SPD server software delivers a fully integrated and seamless way to access large volumes of data and serves large numbers of concurrent users, due to its ability to processes information asynchronously.

For example, SPD 2.0 also provides the ability to execute multiple pass-through SQL statements in parallel:

```
Proc sql;  
  Execute(Begin async operation)  
  Execute(create table one as select * ..)  
  Execute(create table two as select ...)  
  Execute(copy table three .....)  
  Execute(end async operation)
```

SPD 2.0 also provides the ability to do parallel evaluation of select with group by. This is especially useful since it allows a user to perform aggregation just like PROC SUMMARY but SPD 2.0 can do it much quicker.

```
Data = Census PRECS data (2G+)  
Group by – state  
Hardware – 4cpu  
Proc summary - 5min 58 sec  
SPD/SQL - 2min 3sec  
(Statistics provided by SAS website)
```

Another example of using SPD Server to maintain large tables with indexes is shown below.

NOTE: This index creation method is also applicable in Base SAS software only the bitmap indexing option will not be valid.



Release 2.0. The following example demonstrates this technique.

Example 1. Parallel load technique

```
libname spdslib sasspds 'pdata' ....;
libname somelib ....;

/*
* Create an empty SPD Server table with
the same columns and
* column attributes as the existing SAS
table.
*/
data spdslib.cars;
set somelib.cars(obs=0);
run;
/*
* Create indexes for the empty table so the
the indexes
* will be appended in parallel with the table
appends.
*/
proc datasets lib=spdslib;
modify cars(bitindex=(make));
index create make;
index create origin;
index create mpg;
quit;
/*
* Proc append the SAS table cars to SPD
Server table cars.
* The append to the SPD Server table and
its indexes will
* occur in parallel.
*/
proc append base=spdslib.cars
data=somelib.cars; run;
```

(Provided from SAS web site)



Alternatively, if you are using SQL pass-through you should consider using the LOAD TABLE command to perform the same operation. This encapsulates the previous sequence of SAS DATA and PROC steps into an even more powerful technique for gaining maximum performance when loading a new table. The following example demonstrates the same table construction using this technique.

Example 2. SQL LOAD TABLE command

```
proc sql;
connect to sasspds(dbq='pdata' ...);
execute(
libref somelib engopt="..."
) by sasspds;
/*
* Select data to use to create an SPD
Server table
* with indexes
*/
execute(
load table cars[bitindex=(make)]
with index make on(make),
index origin on(origin),
index mpg on(mpg)
as select * from somelib.cars;
) by sasspds;
```

(Provided from SAS web site)

Indexing With the SPD Server

A significant part of SPD Server development focused on the efficient creation, maintenance, and use of table indexes. Indexing can greatly speed the evaluation of WHERE-clause queries. And, the index can be a source of sort order in the case of BY-clause processing.



Summary

There is no perfect blueprint for developing the perfect data warehouse to provide answers to all reporting solutions, since the warehouse and the enterprise layer will most probably be continually expanding.

Overtime some changes may be required to the warehouse structure as the data volumes increase. However, with the topics discussed in this paper it should give the business a clearer picture of how they expect to deliver information solutions.

Contact Information

Andrew Wilcox
Amadeus Software Ltd
Orchard Farm
Witney Lane
Leafield
OXON
OX28 5PG
England

Telephone +44 (0) 1993 878287
Fax +44 (0) 1993 878042
E-mail Info@amadeus.co.uk
Web Page www.amadeus.co.uk

Copyright Notice

No part of this material may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Amadeus Software Ltd. © Amadeus Software. June 2001. All rights reserved.

Trademark Notice

Microsoft products are registered trademarks of Microsoft Plc, USA. Base SAS Software, SAS/MDDDB Server, SAS/CONNECT, SAS/ACCESS Software, SAS/Enterprise Miner Software, SAS/ Enterprise Reporter Software, SAS Enterprise Guide Software, SAS/SPD server software are registered trademarks of SAS Institute, Cary, NC, USA.