The ABCs of MDDBs

Marge Scerbo, CHPDM/UMBC

Abstract

MDDBs (MultiDimensional Databases) are 'in' and SAS Institute is now offering MDDB software as a new product line. What exactly is an MDDB? Is there a simple explanation? The structure of an MDDB is not as complex as it might seem at first glance. This paper will discuss the structure of MDDBs as well as their uses. Access options and functionality will be covered. In addition, a short, personal overview of the pros and cons of this storage technique will be offered.

Introduction

In the world today, computer professionals are surrounded by acronyms: OLAP, OTAP, ODBC, MDDB, DSS ... . It's difficult to attend a conference or a meeting and not be overwhelmed with the influx of new concepts and ever-changing technology. One recent innovation is the availability of MDDBs. An MDDB is a **M**ulti**D**imensional **D**ata**B**ase, which sounds even worse than its acronym. This paper will provide an insight into MDDBs and will hopefully lead to a greater understanding of this promising technique for data storage and information retrieval.

Proc Summary

Many SAS[R] programmers are familiar with Proc Summary, a powerful tool for summarizing and classifying data. Data accessed by a Proc Summary must be in the form of a SAS data set or a SAS data view. The VAR statement defines the fields to be analyzed; these of course must be in numeric format and are usually continuous. A CLASS variable is a method for categorization of the data and usually has a discrete and definable number of values. It allows for identification of specific categories and provides a means to subset the data. The CLASS statement defines the fields which are to be used as classification variables, such as gender, race, or county. A sample Proc Summary statement would be:

```
PROC SUMMARY DATA = ssd.people;
        VAR pay;
        CLASS gender race county;
        OUTPUT OUT = people2
                SUM = clspay;
RUN;
```

In processing these statements, SAS will first create an NWAY table, a table of all possible crossings. For example, in the data to be analyzed, gender has 2 values, race has 5 values and county has 10 values. Therefore, the NWAY table will contain 193 cells, each which contain the resulting sum of the PAY variable, stored as a new variable CLSPAY, for the 500,000 observations. From this table, the _TYPE_ variable will be created. The _TYPE_ variable when equal to 0 will always be contain the total amount of PAY for all observations. The values of the _TYPE_ variable will have the following values:

TYPE	Totals for:	# of Cells
0	NWAY Table	1
1	Each County	10
2	Each Race	5
3	Each County by Each Race	50
4	Each Gender	2
5	Each County by Each Gender	20
6	Each Race by Each Gender	10
7	Each County by Each Gender by Each Race	100

The size of the final SAS file created by the Summary procedure is dependent on many factors: the number of class variables, the number of values/class variable, and the number of analysis variables. By utilizing the value of the _TYPE_ variable, access to the totals in each crossing is available. If a report has been requested which contains the payments by race and county, selecting records WHERE _TYPE_ = 3 will provide the information needed.

MDDBs

An MDDB creates storage for data in a summarized format which provides fast and easy access. The use of a multidimensional database gives the user or the client multiple lines of access, 'dimensions', to the data and summarization by each of these dimensions. This warehouse model can be viewed as a cube with multiple dimensions: a series of cubes creating one large cube. Users can interpret the data from any one of these dimensions or cubes. This design methodology will store aggregates at each dimensional crossing, called cells. A 'cell' is a unique combination of each dimension's level and will contain the summary values for that crossing.

MDDBs provide the ability to examine large amounts of data with great speed. This speed is due in part to the technical design of the MDDB. The SAS/MDDB[R] product contains fast indexes to each subtable as well as to the detail level data. Speed is also accomplished by presenting users with summary level data, rather than all the records which created the summary. Rarely do end users need to look at all data for the entire population; usually views of summarized data which can be subset and/or drilled down are much more useful. A SAS MDDB is a read only file.

Within a SAS MDDB, a summary structure known as an NWAY table is created to store the data. This NWAY table, just as in a summary table, represents a full list of crossings of all class variables named. Summary crossings of specific dimensions can also be created. These crossings or subtables improve access time and therefore efficiency. These summary

crossings, 'subtables', should represent those queries that will be submitted by users to the MDDB. If no subtable exists, the access method will either use the NWAY table or some other subtable which suits the query.

SAS Proc MDDB follows similar implementation to that of Proc Summary. The NWAY table, a full listing of all crossings of the CLASS variables, is created first. This process is memory intensive; in Version 6.12, memory to create the entire NWAY table must be available. Depending on the number of subtables stored, the MDDB may require less disk space then the underlying detail table. The number of statistics stored will also effect the size. The file extension of an MDDB is SSM (Windows) or SSM01 (Unix).

An MDDB is a software implementation that stores summarized data in a manner allowing fast and easy access by some SAS products. Proc MDDB is the SAS procedure that creates an MDDB. The only required statement in a Proc MDDB is the OUT = statement. If no input data set is named, the _LAST_ data set is used. Good programming practices include the inclusion of a specific data set name. Each MDDB contains a minimum of an NWAY summary table, plus any defined subtables which can be derived from the NWAY table. Classification (CLASS) variables can be either character or numeric; analysis variables (VAR) must be numeric. As with many SAS procedures, Proc MDDB can be executed batch or online. A simple example is:

```
PROC MDDB DATA = ssd.people OUT = ssm.peoplem;
        CLASS gender race county;
        VAR pay / SUM;
RUN;
```

What is a hierarchy? A hierarchy is a subtable which allows for quicker and easier access to the summarized data stored in that particular cell. Each hierarchy defined is stored within the MDDB in addition to the large NWAY table. Hierarchies are also called subtables or dimensions. By default, hierarchies are non-display; these hierarchies are not of use when registering the MDDB to a SAS/EIS software in the metabase definition process. Displayed hierarchies (DISPLAY = YES) are automatically recognized in the metabase registration process. These display hierarchies may serve as drill-down access to the data; for example, county would drill down to zip code. Naming hierarchies is useful for ease of definition of the viewed subtable. Unnamed hierarchies will be identified as HIERn. Non-display hierarchies do play an important role when considering ease and speed of use. Any pre-defined hierarchy may be used in place of the entire NWAY table when queries are submitted to an MDDB. Imagine accessing data stored in an NWAY table built on 5 million records, 20 classification variables and 15 analysis variables!

A SAS MDDB has no limitations on the number of subtables or classification or analysis variables. There is a maximum of 8 stored statistics: N, SUM, SUMWGT, UWSUM**,** NMISS, USS, MIN, MAX. In addition, 13 other statistics are available at run time: AVG (Average), RANGE (Difference between Min and Max values), CSS (Correct Sum of Squares), VAR (Variance), STD (Standard Deviation), STDERR (Standard Error of Mean), CV (Coefficient of Variation), T (T value), PRT (Probability of greater absolute value), LCLM (Lower Confidence Limit), UCLM (Upper Confidence Limit), PCTN, PCTSUM. These statistics are dependent on which statistics have been stored. Only cells containing data are stored; the table is sparse and therefore only non-missing data cells are stored.

Design

The design of the underlying detail table and the subsequent MDDB(s) built from it is of first importance. As in other aspects of programming and analysis, the correct design of the data files is imperative and time consuming. Data stored in the detail tables are normally constructed from existing files which have been validated, cleansed, reorganized and sometimes summarized. Some of the access methods to MDDBs allow for reach through to this detail or base table. If the structure of this table is invalid, then the MDDB created from it will also be invalid. MDDBs allow for easier access and analysis; they do not correct inaccuracies. The actual design time of an MDDB is much much shorter than the design of the underlying structure. In the planning of a data warehouse, a large amount of time should be designated to the design of the underlying base tables.

In addition to the design of the detail tables, another design issue is that of subtables. These hierarchies are of utmost importance if the MDDB is to be accessible by users or clients from EIS objects. If a user submits a query to a large MDDB which does not contain a subtable corresponding to the request, the NWAY table will be accessed and subset. Therefore, it is important to include hierarchy statements in the MDDB code for each probable dimension queried.

With SAS Institute's Warehouse Administrator® or similar code generators, MDDB code is created by crossing every single CLASS variable with every other one. In the example above where there are only 3 class variables and a small number of values for each, the code generated and the number of hierarchies included are manageable. In an environment where there are 20 class variables, the number of crossings is greatly magnified. Another consideration is the 2 gigabyte file size limit for any SAS file, including an MDDB. The creation of hierarchies which are pertinent to the users is at first time-consuming but will lead to a more editable number of lines of code and a more efficient use of disk space, memory and processing time. Unfortunately, if additional subtables are needed, the MDDB must be rebuilt.

The first example in this paper of MDDB code did not contain hierarchies. Hierarchy statements are formatted in the following manner:

```
HIERARCHY classvar1 classvar2 ... classvarn
    /NAME = name|'name' DISPLAY = yes|no;
```

where the hierarchy name can be a string of characters with no spaces included or a string with spaces or blanks enclosed in quotes. As stated, the default display setting is no; if this hierarchy will be displayed as a drill-down dimension, display should be set to yes. The order of hierarchy statements is only important in two situations. The first display hierarchy named will by default appear on a displayed table. Second, during the creation of an MDDB, the process will be made more efficient if the subtable containing the most fields be created first. For example, YEAR QUARTER MONTH, then QUARTER MONTH. When creating hierarchy statements, it is important to remember hierarchy-hierarchy crossings. For example, a

report might cross a time hierarchy (YEAR, QUARTER, MONTH) with a geographic hierarchy (COUNTY, ZIP) and therefore this new hierarchy should be included (YEAR, QUARTER, MONTH, COUNTY, ZIP).

Below is a more realistic portion of code, with the hierarchy statements multiplied as needed. MDDB code with multiple analysis and classification variables and several hierarchies may appear as:

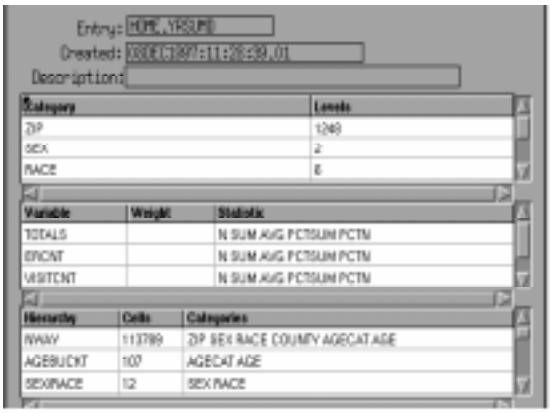
```
Proc MDDB DATA = ssd.people OUT = ssm.peoplem;
CLASS county zip race gender age agecat      .....;
VAR totals/sum n inpttot/sum visits/sum ...;
HIERARCHY agect age /NAME='Agect/Age' DISPLAY=yes;
HIERARCHY county zip /NAME='Cnty/Zip'    DISPLAY=yes;
.....
HIERARCHY county race/NAME='Cnty/Race' DISPLAY=no;
HIERARCHY county age /NAME='Cnty/Age'   DISPLAY= no;
RUN;
```

Accessing Information on MDDBs

The MDDB View Window is available in Version 6.12 Windows and Unix platforms. This window displays the structure of this special SAS data file. This display allows browse access; information on the MDDB cannot be changed. It can be accessed by selecting the DIR (directory) window and placing S for 'Select' before the MDDB to be viewed. It looks similar to a Proc Contents of a data set. Typing MDDB mddbname on the command line will also produce this display.
In Version 6.12 there is a limitation with the usage of this window. If there are any hierarchies which have been named with a description which is over 8 characters, the window will not display any contents and there will be scl errors in the LOG window. According to SAS Institute, this will be corrected in Version 7.

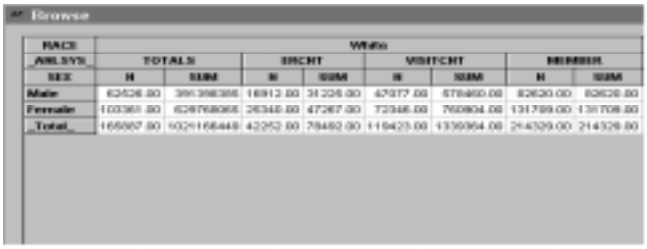
The MDDB Browse window is accessible by entering L (listing) from the DIR (directory) window. This window will display the first hierarchy or subtable in the MDDB. If no hierarchy has been created, the NWAY table will be displayed.
By holding down the right mouse button, a window will appear which will provide a list of hierarchies for selection. By choosing one, a new hierarchy will be displayed.

EIS Access to MDDBs

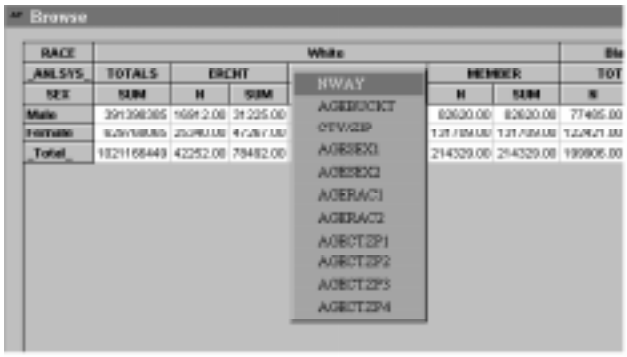


By far the most usable access methods are found in SAS/EIS[R]. There are several EIS objects that can present information stored in an MDDB. These objects include multidimensional tables, business graphs and maps, and organizational charts.

Before an EIS object can be built, the MDDB must be registered to a metabase. A metabase is an indexed SAS data set containing data about the data set and its fields; it does not contain the values stored in those fields. Registering a SAS data set is a multi-step process whereby the data set is



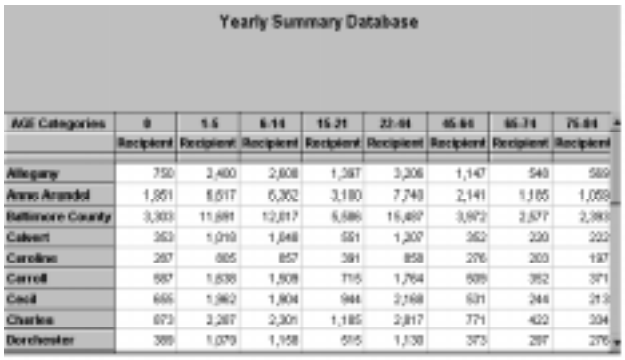
registered and then each table and column attribute is defined. In addition, each hierarchy must be registered. Therefore,



each analysis and classification field must be defined as well as each hierarchy. When registering an MDDB, the process is simpler. To register an MDDB, select Add from the EIS File pmenu and then select the MDDB to be registered. Once the MDDB is registered, each column is automatically included in the metabase. Classification variables are registered as such, as are analysis variables. Hierarchies which have been defined as Display = Yes are automatically registered and can be used as drilldowns. Customization is of course still available, but the registration process is fairly simple. Adding mapping information to the metabase, whether in the case of a data set or an MDDB, is tedious and unfortunately this information cannot be copied from one entry to another.

The most powerful EIS object is the multidimensional table. This table displays the default hierarchy selected. If there is drilldown capability defined, it is automatically available.

In addition, a multitude of options is offered by pressing the right mouse button. The report layout can be changed; across, down and analysis variables can be selected from the variables registered in the metabase. Fields can be customized as to appearance, titles, formats, ranges, and more. It is possible for a user to compute a new field from fields in the metabase as well as constant values. Statistics can be changed or the table rotated. The MDDB can be subset by dimensions; the MDDB/EIS process is intelligent and offers only valid selections. For instance, if the table has been drilled down to a certain age, only the fields which are available for that subset are displayed, as well as the

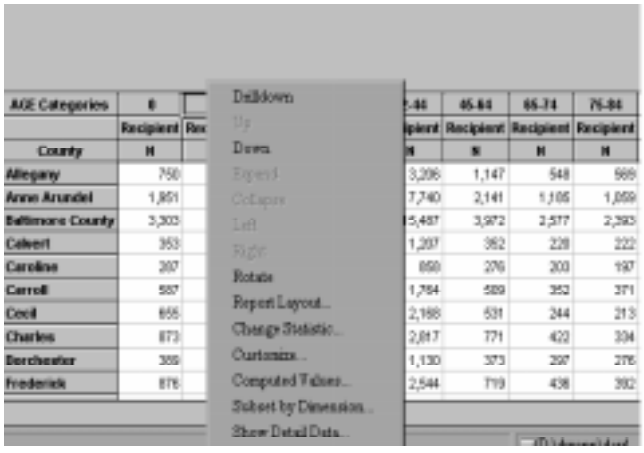


appropriate values for each field. From a specific cell, it is possible to reach through directly to the corresponding subset of the detail table. At any point, it is possible to save the displayed hierarchy or portion of the detail table.

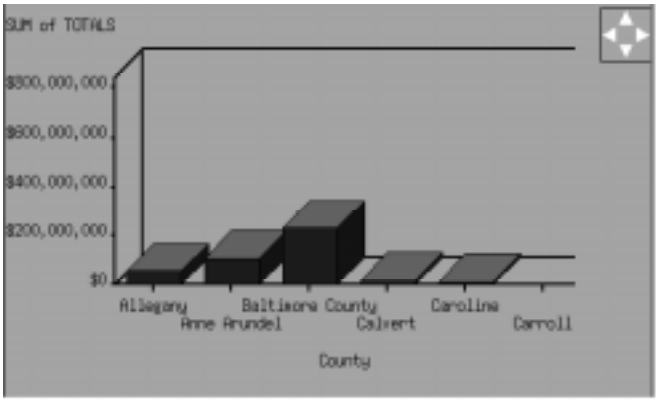
Additionally, graphics objects are available through EIS:

WEB Access

MDDBs can also be easily accessed via SAS Institute's new IntrNet[R] software. Once an MDDB has been created and its metadata registered through SAS/EIS, the mechanism to allow web access is fairly easy. SAS IntrNet software is an application dispatcher and the MDDB Report Viewer is an application within the IntrNet dispatcher application. Very little HTML code needs to be written to make use of this tool. It is easy to implement and use.



The MDDB Report Viewer allows users access to SAS MDDB files. These users do not have to have access to SAS or may



not want to run SAS. Since the EIS registration process defines the MDDB metadata, all fields within the MDDB are available to users' selections as down, across or analysis variables. The report viewer then interactively generates the report defined by the user via the Web. Data that is stored in an MDDB is now more accessible in a very easy fashion.

Pros and Cons to MDDB Usage

There is no one perfect solution to data warehousing. Each method has its pluses and minuses. MDDBs provide a storage mechanism for data which meets many of the needs of users. By correctly designing both the detail table and the MDDB structure, a user can have multiple lines of access into the data. Since the data are usually in some summarized format, it may be difficult to extract actual counts. But although this is a drawback, it does not lessen the effectiveness of this method.

Many users or clients are comfortable with prewritten reports. These reports allow for little or no individual specification or change. If a user does have access to an online system, large amounts of transactional data may be available but any manipulation of this data may be time-consuming or difficult or both. In addition, if manipulation is possible and not controlled, different directions or methods of analysis might produce varying outcome. Which outcome is correct? Although each individual can run varying analyses on data stored in an MDDB, the outcome of each study should be alike since the summarization within each cell is the same. This should prevent different outcomes for the same study no matter which direction was used to navigate the MDDB.

The ease of design of an EIS front end to an MDDB is another plus. Registering an MDDB to a metabase is fairly simple. Updating or 'synchronizing' an MDDB which has had variables or hierarchies added is also not difficult, although not readily apparent. Registration of map information is tedious, and hopefully in future releases of EIS, the method will be modified to allow a programmer to copy this information from one MDDB to another. Creation of EIS objects is not difficult and manipulation of these objects by clients is a great plus. There is no need to design a multidimensional table for each user; each person can in fact do this individually with the pull down menus. EIS software has been enhanced over the years since its initial distribution, and its access to MDDBs will surely increase its usability again.

Summary files are often used in place of an MDDB. From certain perspectives, these files fulfill all the requirements in a

particular setting and are certainly the less expensive option. Summary files are more time consuming to register in a metabase and may require more work in creation of EIS objects to interface with them. In addition, access times are much greater; a comparison of MDDB access to summary table access was run on a Sun/Solaris Unix machine. These two files were created from the same detail table with approximately 600,000 observations and 32 variables. Initial display of the summary table was, over several attempts, at a minimum of 3 times longer than access to the MDDB. Drilldown from one level to another (county to zip) was up to 10 times longer. Report layout changes and subsetting were similarly quicker to the MDDB than to the summary data set.

Another feature associated with MDDBs is 'drip feeding'. As updates to the data are made, the drip feed process will rebuild only those cells which are affected by the update rather than rebuilding the entire MDDB. This process does require at least twice the disk space for the MDDB; a new copy of the MDDB is made in the process and the original copy remains in place.

Two major 'cons' of MDDB are memory usage and cost. In Version 6.12, when an MDDB is created, the first process executed is the creation of the NWAY table. This process is memory intensive; all possible memory is used, depending on the size of this table. If the detail table is large and the memory available is limited, the creation of the MDDB may fail.

MDDB software is separately priced software; it is not bundled into a 'warehouse' cost and therefore, depending on the size of the machine, price can be prohibitive. SAS sales representatives state that there are no plans in the future to include the price of the MDDB product in other product groups which could lower its overall cost.

New with Version 7

As this paper is being written, so are developers at SAS Institute working on Version 7. Some of those changes and updates are described below, although since Version 7 is a work in progress at this time, this discussion should not be taken as gospel.

There are several important features in SAS Version 7 which affect MDDBs. Many of these changes provide the system administrator greater control over the MDDB and its access. HOLAP, Hybrid Online Analytical Processing, will provide greater speed of use of the MDDB. Basically, HOLAP turns the server into a compute server. Subsetting of the MDDB from an EIS object will take place on the server rather than the client, one process which slows the access mechanism. By improving the speed, the use of this important access mechanism will be enhanced. In the past, the subsetting took place on the client, therefore a Windows-based application would attempt to subset the MDDB on the pc itself and not on the remote server, quite a slow process.

HOLAP will also include a redefinition of MDDBs and their subtables. In Version 6.12 an MDDB was a single file; in Version 7 an MDDB can be a collection of files. If a summary NWAY table already exists, rather than recreating this data in an MDDB, the registration of the MDDB to the metabase will define this data set as the NWAY table. Subtables which were previously stored in MDDBs might in the future be SAS data sets, SAS data views, relational database files. These subtables will be defined as such to the metabase. Accordingly, a file can be defined to multiple MDDBs, not just exist for one.

The developers have also been working with the memory use during the creation of an MDDB. This process can be a 'memory hog', especially with a large MDDB. In Version 6.12 it was necessary to have enough memory available to create the largest possible subtable, usually the NWAY table. Memory usage will be paged out; this may cause the creation process to take more time but use less memory.

MDDBs will also be smaller. Developers are working to redesign the internal structure and the indexes. Therefore, MDDBs will require less disk space. Also, since one subtable may be used by multiple MDDBs, the disk usage may decline. At the same time, as the MDDBs grow smaller although creation time might be increased due to paging, the decrease in the file size may help defer the increased creation time.

Very important changes within HOLAP include those of security. In Version 6.12 in order to have various access levels to the summarized data, different MDDBs had to be created for each security level. In Version 7.0 the system administrator can set security levels for each summarized level of an MDDB. For example, a county health officer might only have access to county level data and totals, whereas a Department of Health official may have access to much greater level of granularity.

There are new EIS objects that will be available in the new version, including a forecasting object. Additional chart types, including a bubble chart, have been added to the EIS software which can display MDDB data. Version 7 will also include more customizing options in the application access to an MDDB, but all custom code from Version 6.12 will be carried forward and usable by Version 7.

Conclusion

This section will contain a personal view of MDDBs from my study and usage of them. I work in a data-laden department; seven years of Maryland Medicaid claims totals to many many millions of records. One year might have over 30 millions records. In the past when a health care or policy analyst needed analysis, the process required a programmer as well as possible intensive machine use. About a year ago we undertook the task of designing and developing a data mart. This was done in a team setting with the consultants from SAS Institute and Price Waterhouse.

Time was spent interviewing users to define their needs. Of course, this was a difficult task since personnel could not readily define their needs. This is not a production environment where monthly or quarterly reports satisfy the users and their studies. This is a more research style organization where each project may have very individual specifications. So interpreting the needs of the users to a clear data model was an ongoing process. By far the most time consuming and mind bending portion of the process was the design of the base tables. Anyone who is undertaking the development of a data warehouse should take this to heart; the detail tables are the core to the warehouse. Additionally, the selection of hierarchies, especially non-display subtables, was an ongoing process. Our design included at least 20 class variables for each MDDB. Needless to say, the crossing of all those fields produced by code generators was huge, with up to 500,000

lines of hierarchy statements. Therefore, we had to decide which subtables were the most likely to be studied, with the understanding that additional hierarchies might need to be added in the future. Unfortunately, this will require an entire rebuild of the specific MDDB. In our case, it was often apparent that the build of the detail table was in fact more time consuming than the accompanying MDDB.

It might seem like there were many negatives to the use of MDDBs but they are by far outweighed by the positives. Our users can now access the data and run exploratory analyses on their own. This may not become the final outcome of a study, but it means that studies which our data cannot support or for which there is not a reasonable population to study, are not submitted to the already over-tasked programmers. A specially built search screen which actually subsets the detail table and creates an ad hoc temporary MDDB will become a handy study tool for the analysts.

One of the drawbacks is speed, especially of the subset screen. With the inception of Version 7, I see that the use of this data mart will increase many fold. Security issues, imperative to the confidentiality of our data, will also be handled with the new version. It will be interesting to see how much our present design can be reworked when the new structure of MDDBs allowing for the inclusion of a variety of files is available.

In closing, I doubt that the development of this data mart could have been accomplished in the short time without the availability of the MDDB product. The learning curve is fairly short, once the design is complete. I look forward to future data warehousing projects!

Bibliography

Barnes Nelson, G. (1996) 'An Introduction to Data Warehousing', NorthEast SAS Users Group Conference 96 Sunday Workshop

McIntyre, J. (1996), 'Multidimensional Data Model Extensions to Data Warehouses', Proceedings of the Twenty first Annual SAS Users Group International Conference, 21

Special thanks to SAS Institute employees, Stuart Levine, Joe Costanzo and Fritz Lehman for their patience and assistance on this paper.

Marge can be contacted at:

Marge Scerbo
CHPDM/UMBC
1000 Hilltop Circle, SS Room 309
Catonsville, MD 21250
scerbo@umbc.edu