Paper 121

The ABCs of MDDBs

Marge Scerbo, CHPDM/UMBC

Abstract

MultiDimensional Databases, affectionately called MDDBs, are powerful storage mechanisms used in data warehousing. These file structures are available with SAS MDDB Server software. When combined with multidimensional objects available in SAS/EIS software, information can be made accessible to analysts quickly, easily and correctly. This paper will review the basics of MDDBs presented last year. These topics will include structure and design of MDDBs as well as coding tips. Further discussions of updates to the MDDB Server and SAS/EIS Software in Version 6.12 and additions found in Version 7 will be included. In particular, HOLAP (Hybrid Online Access Processing) will be discussed. HOLAP attributes decrease access time and have added many important options.

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® 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 198 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 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.

Storage and Access of Data

When discussing storage mechanisms, it is also important to include different methods to access these files. Without the ability to analyze and report data stored in a file easily and efficiently, the file may be useless. This paper will discuss the storage of data in a SAS MDDB structure and access to this data using SAS/EIS software, with and without the new HOLAP extensions. For further information on other access mechanisms, specifically those that involve batch processing through data steps and procedures, check previous SUGI and NESUG proceedings. These papers include specific instructions on creating a LIBNAME statement with an SASSFIO extension which will point directly to a specific dimension within an MDDB.

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 a cell. A 'cell' is a unique combination of each dimension's level (across and down) and will contain the summary value 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 Server [R] software 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 or provide a method to look through or drill down to different layers of information. These summary crossings, 'subtables', should represent those queries that will be submitted by users to the MDDB. If no subtable exists, the access method may 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. Update 045 for Version 6.12 of SAS for Unix machines will solve some possible problems caused by the size of the NWAY table and its creation; this update allows for the storage of SAS files over the former 2 gigabyte limit. Depending on the number of subtables stored and the number of class and analysis variables used, the MDDB may require less disk space then the underlying detail table. In cases where there are several class and analysis fields and very little of the data are missing, the size of the MDDB may actually be larger than the base table. The number of statistics stored will also effect the size. The file extension of an MDDB is SM2 (Windows) or SSM01 (Unix).

Size of MDDBs

Often times, management is interested in the size of files and the amount of disk space which may be used for these files. SAS Institute has created an algorithm to calculate the approximate size of an MDDB. This algorithm is also available from SAS Institute as a macro program to allow a calculation mechanism for the sizes of multiple MDDBs. The sparsity of the data has a great affect on the final size.

For every:	Add in overhead	Plus
MDDB	900 bytes	
Analysis Variable	676 bytes	
Class Variable	340 bytes	(max formatted length * number of values) + (unformatted length * number of values)
Hierarchy	296 bytes	(number of dimensions * 4 + number of analysis vars * number of stats * 8) * number of crossings

For an example of SAS data sets which have been used as detail tables for the development of MDDBs, here are tables showing the number of records, the number of class variables, analysis variables and hierarchies, and the file sizes of each data set and corresponding MDDB on a Unix machine:

File	# of Records	Class Fields	Analysis Fields	# of Hierarchies
Diagnostic	3,814,792	24	8	6
Nursing	74,183	24	15	6
Pharmacy	9,985,250	26	5	6
Physician	7,143,687	27	8	8
Surgical	3,814,792	23	3	6
Yearly	1,179,639	23	28	5

File	Detail File Size	MDDB Size	% Diff
Diagnostic	1,328,267,264	680,615,936	49%
Nursing	19,423,232	11,714,560	49%
Pharmacy	2,115,510,272	1,148,264,448	46%
Physician	1,721,237,504	821,657,600	52%
Surgical	66,560,000	15,990,784	76%
Yearly	424,878,080	308,158,664	27%

Clearly, although there are a great number of class fields as well as several analysis fields and hierarchies, the size of the MDDB in each case is much smaller than its corresponding detail data set.

Proc MDDB

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. Although it is possible to use numeric fields as

class variables, SAS will convert these fields to character in BEST12 format. If this is the case, updates to the MDDB may be difficult. It is best to convert all class variables to character in the underlying base files. In addition, there is no FORMAT statement allowed within the MDDB procedure. Therefore, format all fields, both numeric and character, prior to the creation of the MDDB. 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 may play an important role when considering ease and speed of use, depending upon the access mechanism. A 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. Dependent upon which statistics have been stored, up to 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. 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 issue is that of subtables: how many subtables and which ones? These hierarchies are of utmost importance if the MDDB is to be accessible by users or clients from EIS objects when HOLAP extensions are not available or not used. 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. Yet, there is no reason to create every possible subtable.

With SAS Institute's Warehouse Administrator® or similar code generators, MDDB code is created by crossing every single CLASS variable with every other one and every combination of those variables. 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. 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. Keep in mind that 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:

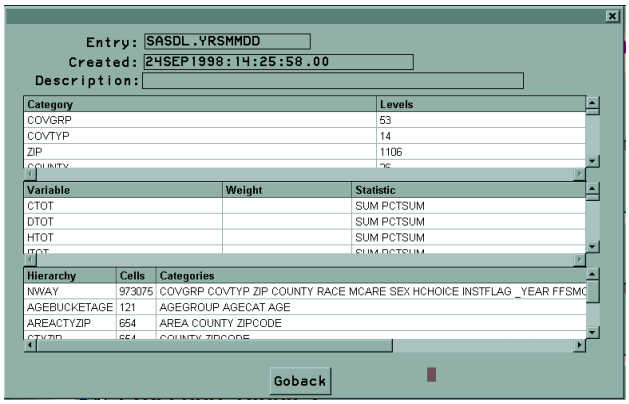
```
HIERARCHY classvar1 classvar2 ... classvarn
     /NAME = name|'name' DISPLAY = yes|no;
```

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 ;
VAR inpttot visits/sum ...;
HIERARCHY agecat age /NAME='Agecat/Age'
DISPLAY=yes;
HIERARCHY county zip /NAME='Cnty/Zip'    DISPLAY=yes;
.....
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. When working within a windowed environment, double clicking on the selected MDDB will also display this screen:
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 contains certain special characters, the window will not display any contents and there will be SCL (Screen Control Language) errors in the



LOG window. According to SAS Institute, this will be corrected in Version 7.

EIS Access to MDDBs

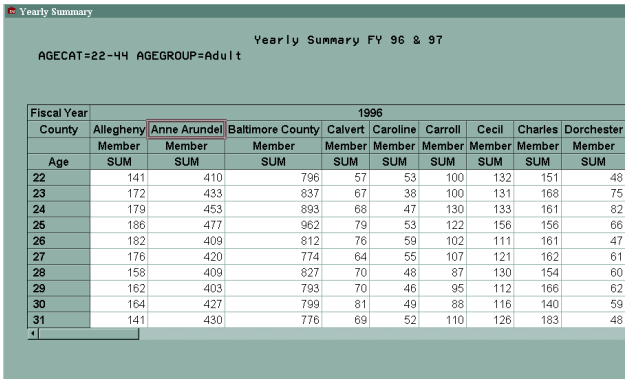
By far the most usable access methods are found in SAS/EIS[R]. SAS/EIS is a separately licensed software package. There are several EIS objects that can present information stored in an MDDB. These objects include multidimensional tables, business graphs and maps, and organizational charts.

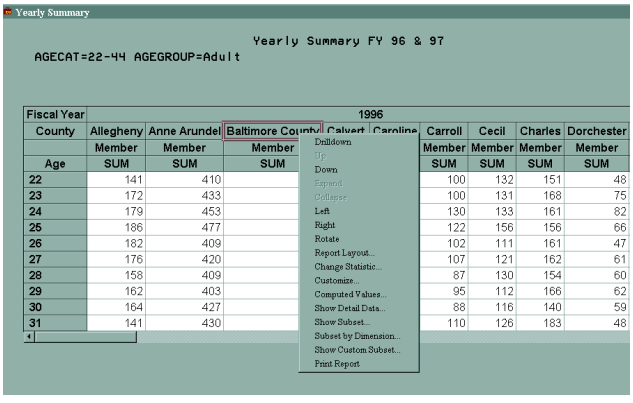
Before an EIS object can be built, the MDDB must be registered to a metabase. A metabase is an indexed SAS data set containing data about the data set and its fields; it does not contain the values stored in those fields. Registering a SAS data set is a multi-step process whereby the data set is registered 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. First, select the **METABASE** icon from the EIS menu screen. 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 drill downs. 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 next step in the process is to Create an Object, a mechanism for displaying the actual information stored in a SAS MDDB. Begin by clicking on the **BUILD EIS** icon from the EIS menu. Then **Add** the object desired.

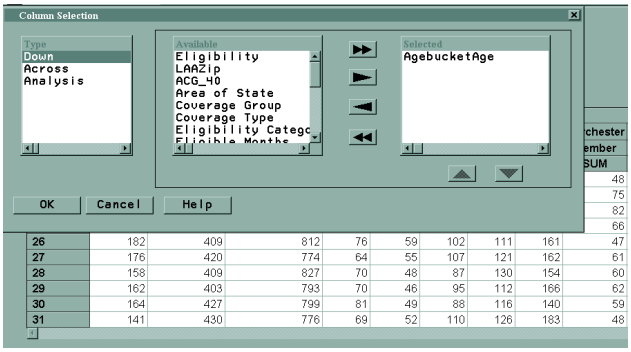
The most powerful EIS object is the multidimensional table. This table displays the default hierarchy selected. If there is drill down capability defined, it is automatically available. An example of a 'MultiDim' is:
In addition, a multitude of options is offered by pressing the right mouse button.





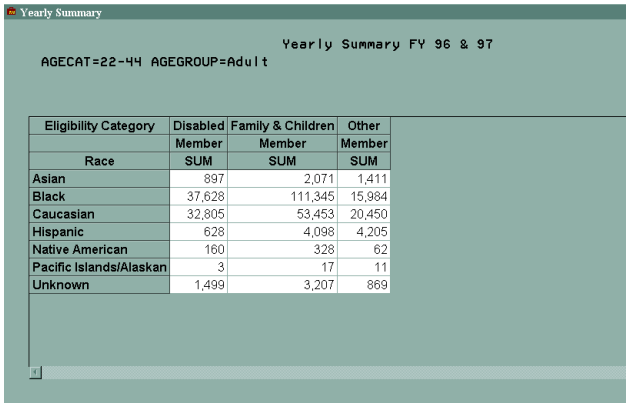
The report layout can be changed; across, down and analysis variables can be selected from the variables registered in the metabase. Fields can be customized as to appearance, titles, formats, ranges, and more. Totals can be added to rows and/or columns. It is possible for a user to compute a new field from fields in the metabase as well as constant values. Statistics can be changed or added. The table can be rotated. The MDDB can be subset by dimension; 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.

An example of the report layout screen is:
The user has complete control, within the boundaries of the



defined metadata, of what appears in the final report. Here is an example of a user defined report, created from the original report:

There are times when the structure of an MDDB has been



Eligibility Category	Disabled Member	Family & Children Member	Other Member
Race	SUM	SUM	SUM
Asian	897	2,071	1,411
Black	37,628	111,345	15,984
Caucasian	32,805	53,453	20,450
Hispanic	628	4,098	4,205
Native American	160	328	62
Pacific Islands/Alaskan	3	17	11
Unknown	1,499	3,207	869

changed. In order to update the Metabase to these changes, you need to 'Synchronize' the Metabase. Again, click on the SAS/EIS icon **Metabase**. Select the MDDB in the metabase to be updated and then highlight the **MDDB Attribute,** click on Edit, and **Synchronize the Metabase**.

HOLAP Extensions

HOLAP, Hybrid Online Analytical Processing, was originally developed by SAS Institute in Europe. Some portions of HOLAP have been made available, on a limited basis, in Version 6.12. HOLAP provides increased functionality for MDDBs and their access.

HOLAP extensions were written to address performance and scaling problems when using SAS/EIS and MDDB Server software in a client server environment. HOLAP is an optimization process for OLAP applications built with SAS/EIS Version 6.12. Additionally, HOLAP extensions allow the use of any number of data sources on multiple platforms to form a 'virtual cube'. This cube can then be used as input to any SAS/EIS report that is multidimensional. Large crossings can be stored in individual summarized SAS data sets (one crossing per data set), rather than all crossings being stored in one large file. SAS/ACCESS can be used to access already summarized and stored in a DBMS without loading it into a SAS data set or MDDB.

The main features of HOLAP which will be discussed in this paper include:

- Transparent compute server processing to improve performance. In short, the HOLAP extensions generate SAS code locally when a request is made. This code is then submitted to the remote server where subsetting and rollups are processed, and the result is read into a new and much smaller MDDB. Without HOLAP, EIS would provide standard method where all subsets and rollups are done on the client after moving the entire crossing down from the server.
- The use of distributed data to increase scalability.
- Logging of requests to analyze usage and provide a mechanism to assist with changes which might be necessary.

There are additional features of HOLAP which will not be covered in detail in this paper:

- Caching for quicker response to resubmitted requests.
- Improvement of object initialization times.

By turning the server into a compute server, HOLAP greatly increases the speed in the use of the MDDB. The subset option of an EIS object, 'Subset by dimension', is normally processed on the client, therefore a Windows-based application remotely accessing MDDBs on a more powerful platform would attempt to subset the MDDB on the pc and not on the remote server, quite a slow process. The HOLAP addition to this object's function redefines the subsetting to occur on the server. In-house tests of this option proved subsetting times were decreased exponentially; the process time dropped from 20 minutes to 10 seconds on one MDDB. By improving the speed, the use of this important analysis tool will increase among the users.

Here is a step by step walk through of the HOLAP process for Release 1.0 TS115 as of November 3, 1998 (operations on *pc are shown in italics* and on **unix in bold**):

1- *HOLAP executes a Proc Upload of a work catalog _DPMC_M which contains an SLIST entry and an SCL entry. The SCL list (SLIST entry) contains information about the classes or dimensions needed to satisfy the data request.*

2- **The information contained in the SLIST is used to build the CLASS statements in the Proc MDDB that HOLAP builds. This program reads**

the permanent MDDB, passes the information from the list to extract the proper cells, executes the appropriate subsetting contained within the passed WHERE clause, and builds the WORK._DATA001 data set on the remote system. This process is run in batch mode. The HOLAP log will show which hierarchy has been chosen and the number of cells in that hierarchy.

3- A macro variable containing the name of the catalog is created and a Proc Catalog is run to delete the original SLIST.

4- Proc MDDB is executed and _MDDB001 is created from the data set _DATA001. The class variables had been stored in the SLIST entry and are now the only dimension variables in the new MDDB. As a default, HOLAP builds the new MDDB with all analysis variables in the original MDDB. The statistic is the single statistic stored in the original MDDB; it is read from the MDDB when the SCL entry was executed.

5- Then this new much smaller MDDB is downloaded to the pc. The HOLAP log contains the number of records and the number of bytes downloaded.

6- After the MDDB has been downloaded to the pc, the HOLAP processes executes a Proc Datasets to delete the new MDDB _MDDB001 and the new data set _DATA001. This cleans up the WORK library.

7- Once this process is complete, the local HOLAP classes replace the permanent MDDB defined for the multidimensional report with the temporary one which was just created and downloaded. For example, a basic report contains a Geographic Hierarchy and an Age Hierarchy. HOLAP reads the dimension variable names from the Multidimensional Report (County and Age Group) and places them in the SLIST. These will be read on the remote server to create the data set extracted from the original MDDB, then to build the CLASS statements in the PROC MDDB.

During a drill down, the same process occurs. HOLAP reads that the dimensions needed to satisfy the request are County, Zip, and Age Group, stores that in a new copy of the list, uploads, processes, builds the new MDDB (incrementing the MDDB number from 001 to 002), and downloads the results. These HOLAP extensions provide huge performance improvements for many complex analyses.

HOLAP will also include a redefinition of MDDBs and its subtables. In Version 6.12 an MDDB was a single file; with the addition of HOLAP, an MDDB can be a collection of files, a 'virtual' cube. If a summary NWAY table already exists, rather than recreating this data in the MDDB build, during the EIS metabase registration process this data set may be selected as the NWAY table. Subtables which were previously stored within MDDBs might in the future be SAS data sets, SAS data views, or relational database files. These subtables will be defined as such to the metabase. Accordingly, a file can be defined to multiple MDDBs and not just exist for one MDDB. With HOLAP a virtual MDDB may be defined, not a physical cube but a collection of files.

Data marts and data warehouses that already exist in other formats, such as a star schema, can now be encompassed in a 'virtual' MDDB. Rather than reworking the structures that are already in place, HOLAP's ability to define these structures within the metadata will allow an MDDB to actually be a distributed database. HOLAP will add a great deal of flexibility to the MDDB and its access.

HOLAP tracks the user requests. These requests are stored in a temporary SAS data set in the user's work directory. This file is called DP_M_LOG and is located in the SAS local work directory. The contents of this directory can be extremely useful in tracking the users of the system, what requests have been made, where bottlenecks occurs, etc. SCL code, Screen Code Language, can be written to store this data set in a permanent directory for system administrator usage. The contents of the HOLAP log include:

Variable Name	Type	Label
METHOD	$40	Method
PROPNAME	$17	Proposed Data Name
PROPHIE	$8	Proposed Data Hierarchy
PROPCELL	8	Proposed Cells
PROPCLAS	8	Proposed Classes
ACCTIME	8	Access Time
ACCMETH	$8	Access Method
DATETIME	8	Datetime
CLASSES	$200	Classes
ANALYSIS	$200	Analysis
SUBSETS	$200	Subsets
SERVER	$8	Server Name
SYSVER	8	SAS Version
MEMTYPE	$8	Member Type
DBMSNAME	$8	DBMS Name
CELLTHRE	8	Cell Threshold
SEARCHPR	$17	Search Priority
EXACT	$40	Exact Match?
OBJECT	$40	Object Name
CACHNAME	$17	Cache MDDB Name
USERID	$40	User Identification

HOLAP Installation

To install HOLAP extensions to SAS/EIS software and attach these extensions to both an MDDB and its corresponding object(s) is a multi step process:

1- Request HOLAP Extensions from SAS Institute.
2- Install HOLAP Extensions in SAS/EIS:

HOLAP extensions are installed on the client in the SASTOOL directory (make sure to assign a LIBNAME to the SASTOOL directory). HOLAP is installed by clicking on the SAS/EIS icon **Setup**. Install the extensions in the **Environment Catalog Search Path** (SYSTOOL._DMDDB), the **Attribute Search Path** (HOLAP attributes), and the **Environment Resource Management** (HOLAP Resource).

3–Add HOLAP extensions to the MDDB attributes:

Double click on the SAS/EIS icon **Metabase** Select the specific MDDB in the metabase and then **Add** an attribute. Select **_ASSDATA** (HOLAP Associated Data).

4–Change the model associated with the object:

Select the SAS/EIS icon **Build EIS** and choose the Report or Graph in the application database associated to the MDDB. Edit this object. Click on the **Advanced** Button and replace **Model** default with **SYSTOOL._DMDDB.HOLAP_M.CLASS**. Click on OK and save the changes to the object.

Version 7 Features and Beyond

Beyond HOLAP enhancements, Version 7 will include other new features affecting the creation and use of MDDBs. Under Version 6, creation of MDDBs could often fail for lack of memory. The creation 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. Version 7 will now allow memory usage to be paged out; disk space will be used when virtual memory has been totally utilized. This may cause the creation process to take more time but use less memory.

MDDBs will also be smaller. Developers have worked 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 MDDBs grow smaller and creation time increased due to paging, the decrease in the file size may help defer the increased creation time.

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 for objects accessing an MDDB, but all custom code from Version 6.12 will be carried forward and usable by Version 7.

Beyond Version 7, SAS Institute is developing tools to make MDDBs meet Open API standards for OLAP. Beginning this fall, SAS will begin beta-testing an OLE DB product that will allow Windows software such as Brio and Cognos to directly access the data stored in a SAS MDDB. The business objects of these software products should be able to drill down and navigate an MDDB whether it be on the same platform or across a TCP/IP network without the use of SAS/Connect. This will allow the data to be verified, tested, summarized and stored in SAS MDDB format while accessed by non-SAS products.

Pros and Cons of MDDBs

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 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 may be much greater.

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. The underlying base table must also be updated in this process. The use of the drip feed mechanism may be useful to systems that are updated on a frequent basis. For systems that require large amounts of processing to produce the detail table, this process may not be as useful.

Two negatives associated with an MDDB implementation 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. It is important to schedule the creation of the MDDB for the least busy hours.

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.

Conclusion

SAS MDDB Server software is an incredibly powerful tool, especially when it is coupled with access by SAS/EIS with HOLAP extensions added. There is a vast amount of data collected and stored but unused because of space constraints, a shortage of programmers, minimal data access, poor management direction, inaccurate specifications, etc. Some of these problems can be overcome by the development of a Decision Support System built with SAS Software.

If management is studying the viability of such system, it is important to remember that the most important task is that of correct data base design. This will include an assessment of the needs of users, present and future hardware capabilities, a study of the data which is presently collected. Then comes the ponderous task of sifting through all this information, choosing what requests can be met by such a system (and what cannot be met), prioritizing tasks, and creating a 'package' that can be built upon as needs increase.

If there is a clear understanding of what data are available and a clear goal for the development of a Decision Support System, SAS software indeed provides all the tools necessary for the creation of a useful tool for analysts, managers, researchers, etc. The time for such development is dependent on in-house SAS knowledge and in-house analytic skills.

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, Mark Moorman, 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@chpdm.umbc.edu