

# Getting a Grip on a Growing Concern: Managing Large Data Sets with a Multidimensional Database

Mark Moorman



*Mark Moorman is product manager for Online Analytical Processing (OLAP) in the Business Solutions Division at SAS Institute. His areas of expertise are the OLAP server, SAS/EIS® and SAS/AF® software, and Web enablement. Mark has a BS degree in English from Liberty University and has been a SAS® software user for nine years.*

## Abstract

*With Release 6.12 of the SAS System, SAS Institute will offer new capabilities for multidimensional analysis. This capability will be available in the product SAS/MDDB™ Server, and will be available for use in SAS/EIS software. This article covers the basic designs for building this new database. It covers how the mechanics of a multidimensional database work, and what makes an MDDB (Multidimensional Database) effective for delivering large data fast. The procedure to build MDDB's databases is covered at length. This article also mentions briefly new features of SAS/CONNECT® and SAS/SHARE® for faster access to data across a network.*

## Introduction

With Release 6.08 of the SAS System, SAS Institute introduced SAS/EIS software with "Slice and Dice" capabilities. This powerful new tool allowed users to drill down, up, and around in their data based on predefined hierarchies. This "Slice and Dice" capability allowed customers to unlock valuable pieces of information hidden under a year, a month, or a product group, by comparing them to other years, months, and product groups. Today this functionality has taken on a new buzzword – OLAP (Online Analytical Processing). The limitation to this earlier release of SAS/EIS was

not with its function but with its capability to effectively process large volumes of data. Releases before 6.12 of SAS/EIS required aggregations (or summarization) of data to occur at each execution. This meant that larger data and data on other platforms was really inaccessible due to system and network constraints. With Release 6.12 of the SAS System, SAS Institute announces a new product called SAS/MDDB Server. With this product, it is possible to examine large amounts of data with amazing speed. The SAS/MDDB Server creates a new data store that is specifically designed to access data at dimensional crossings in a matter of seconds.

All SAS MDDB's have the following:

- a filetype of MDDB
- no limitations on the number of subtables
- no limitations on the number of classification variables
- no limitations on the number of analysis variables
- a maximum of 8 stored statistics
- a possible 11 other statistics at run time
- only nonmissing cross-tabular data
- incremental update capability
- a predictable size
- batch and online creation capability.

The SAS/MDDB Server precalculates crossings of dimensions, and stores only that value. This is how the true performance of OLAP is discovered. Instead of looking at data as a million records, you can look at it as five for yearly data in a five year history file, or 12 for monthly data by year, or 20 for products in a product group. Dimensions can be created much like hierarchies. These dimensions are then stored into a data element (which is not the traditional SAS data set). At the base of this structure is an NWAY crossing of all dimensions (for those familiar with the SUMMARY procedure, NWAY is very much like this). This NWAY does not have a value for every crossing that exists, but from this NWAY, any dimension can be calculated. On top of the NWAY structure are subtables that are explicitly defined at build time. There can be as many of these subtables as there are possible dimensional crossings. A subtable is basically the answer to a question. In other words, a subtable has stored in it the exact information requested, and no calculation is needed. On top of this structure sits a very fast index that has pointers to each subtable, and even a pointer to the base table used to create

this MDDB for "Reach Through". The index takes the query and finds a subtable if one exists, or it goes back to the subtable that most closely represents the dimension and calculates the answer. All of this makes for a very fast, tunable storage technology.

There are basically three ways to build an MDDB using the SAS System. You can use a procedure, an EIS Object, or SCL functions. This article discusses in detail the procedure, but the logic used can easily be translated into the other tools. There are some functional differences between using the procedure and the EIS object:

#### The Procedure

- does not require EIS
- does not require the table to be registered in the metabase
- runs on MVS as well as other hosts
- allows naming of subtables.

#### The EIS Object

- uses Point and Click interface
- uses Metabase registration
- has some modeling capabilities to populate subtables.

## The Procedure

The syntax for the procedure is as follows:

```
PROC MDDB <data=dsname> out=lib.mddb
          <in=orig.mddb> <label=description>;
CLASS categorical variable-list </order-options>;
HIERARCHY class variable-list </hierarchy-options>;
VAR analysis variable-list </stat-options>;
RUN;
```

## The PROC MDDB Statement

The only required parameter in the PROC statement is OUT=. There is no default MDDB name. The DATA= option specifies the input data to be summarized. If DATA= is not specified, \_LAST\_ will be used. The IN= option is used for incremental updates and is discussed later. The LABEL= option specifies a description to be stored with the MDDB. The character description string can be up to 80 characters long.

Here is an example:

```
proc mddb data=sashelp.prdsale out=work.mddb;
```

## The CLASS Statement

You can specify one or more CLASS statements. However, a class variable may only appear once in all CLASS statements. The syntax is

```
CLASS categorical variable-list </order-options>;
```

The CLASS statement lists variables chosen to be used as classification variables in the MDDB. Classification variables are values that define an item. These variables can be either numeric or character. There are several options for the CLASS statement, and they are as follows:

- ASCENDING (default) - sort ascending
- DESCENDING - sort descending
- ASCFORMATTED - sort ascending by SAS format
- DESFORMATTED - sort descending by SAS format
- DSORDER - use order in the data set.

Order-options in the CLASS statement decide the order in which the class values will be displayed during reporting. To specify different sort orders for different class variables, use separate class statements.

Here are two examples:

```
class country region division product prodtype;
```

```
class product /ascending;
class prodtype /descending;
class country /ascformatted;
class region /desformatted;
class division /dsorder;
```

The order of the variables in the CLASS statement or the order of the CLASS statements does not matter. Also, the CLASS statement itself can come anytime after the PROC statement.

## The Hierarchy Statement

You can specify zero or more hierarchy statements. If one is not specified, an NWAY hierarchy is produced. You can specify from one to all of the class variables in each HIERARCHY statement, but each class variable can only be specified once per statement.

The syntax is

```
HIERARCHY class variable-list <name=name|
          "name" ><display=YES|NO>;
```

The HIERARCHY statement creates named subtables in the MDDB. These subtables are created to give quicker access to a particular aggregation of the data. Named subtables are by default considered to be nondisplay hierarchies and will not be recognized by SAS/EIS software during the metabase registration process. If you do not explicitly name your subtables, the default is HIER $n$  (where  $n$  = 1 to number of hierarchies requested). Subtable names are not used in the SAS/EIS viewers when exploiting the MDDB. Naming the subtables is mostly for management, however, it is possible to use the SASSFIO engine with MDDB's, and that requires using the subtable name.

*continued on next page*

To specify a name for your subtable, use the NAME= option of the HIERARCHY statement. The name must be embedded in quotes if it contains spaces.

Here are two examples:

```
hierarchy country region division /name=geo;
```

```
hierarchy country region division /name="geographic hierarchy";
```

If you specify DISPLAY=YES (NO is the default), the hierarchy you create will be created as a "display hierarchy" and will be recognized by SAS/EIS during the metabase registration process.

Here is an example:

```
hierarchy country region division /name=geo display=YES;
```

If you specify DISPLAY=YES (NO is the default), the hierarchy you create will be created as a "display hierarchy" and will be recognized by SAS/EIS during the metabase registration process.

The order of the HIERARCHY statements is not important, and the HIERARCHY statements can come anytime after the PROC statement. The order of the variables on the HIERARCHY statement is only important if the hierarchy is being defined as a display hierarchy. If it is defined as a display hierarchy, then the order of the variables on the hierarchy will be the drill-down hierarchy in SAS/EIS.

## The VAR Statement

You can specify one or more VAR statements. An analysis variable may only appear once in all variable statements. If no statistic is specified SUM will be used. The syntax is

```
VAR analysis variable-list </statistical option-list>;
```

The VAR statement lists the variables chosen as analysis variables as well as the statistics to be precalculated and stored in the MDDB. All analysis variables must be numeric.

The statistical options in the VAR statement can be one or more of the following, separated by spaces:

Analysis	Description
N	count of values
SUM	summarization of values
SUMWGT	weighted summarization
UWSUM	unweighted summarization
NMISS	number of missing values
USS	uncorrected sum of squares
MIN	minimum value in category
MAX	maximum value in category
WEIGHT	numeric variable to use as weight against value

If WEIGHT= is specified, its value must be the name of a numeric variable in the data set. If SUMWGT is also specified, it will be stored in the MDDB. If only WEIGHT is specified, the weight will be used in calculating the SUM statistic, but SUMWGT will not be stored, and the other statistics that would be calculated based on SUMWGT will not be calculated. If SUMWGT is specified but not WEIGHT=, then the request to store SUMWGT will be ignored. The previous list of analyses are stored in the MDDB. There are a series of other analyses that can be calculated. The calculated variables and the required stored variables are as follows:

Analysis	Analysis needed to create	Description
AVG	SUM,N	Average
RANGE	MIN,MAX	Difference between minimum and maximum values
CSS	SUM,N,USS	Corrected Sum of Squares
VAR	SUM,N,USS	Variance
STD	SUM,N,USS	Standard deviation
STDERR	SUM,N,USS	Standard error of mean
CV	SUM,N,USS	Coefficient of variation
T	SUM,N,USS	T Value
PRT	SUM,N,USS	Probability of greater absolute value
LCLM	SUM,N,USS	Lower confidence limit
UCLM	SUM,N,USS	Upper confidence limit

An example is

```
var actual / n sum nmiss;  
var predict / n sum nmiss uss min max;
```

## Building an MDDB

To get started, we can use a sample data set in the SASHELP library. The data set PRDSALE is an example of what some dimensional data might look like. There are basically three dimensions in this data set: Time, Geography, and Product. The Time dimension is comprised of the variables YEAR, QUARTER, and MONTH. The Geography dimension is made up of COUNTRY, REGION, and DIVISION. The Product dimension is made up of PRODTYPE and PRODUCT. These dimensions also make up the hierarchies. These data also have two analysis variables, ACTUAL and PREDICT. Using this data, we can take a look at some example code to build an MDDB.

```
proc mddb data=sashelp.prdsale out=sasuser.testmddb;  
  class product / ascending;  
  class prodtype / descending;  
  class country / ascformatted;  
  class year quarter / desformatted;  
  class month / dsorder;  
  hierarchy country / name = "geo+" display = yes;  
  hierarchy year month quarter / name = "time+"  
    display = yes;  
  hierarchy prodtype product / name = "prodtype+"  
    display = yes;  
  var actual / n sum nmiss;  
  var predict / n sum nmiss uss min max;  
run;
```

This example creates an MDDB in the SASUSER library called TESTMDDB. If you are running SAS in interactive mode, you can take a look at this MDDB by using the DIR window in DMS. On the Command Line, enter DIR WORK. This will produce a listing of all the SAS files in the WORK library. You should see a file called TESTMDDB with a MEMTYPE of MDDB. This is the MDDB that was just created. By placing an 'S' in the space before the name, you will produce a window that describes this MDDB. This special utility window allows you to see the following:

- classification variables and number of values
- analysis variables and associated statistics
- hierarchies(subtables), number of cells (crossings), and class variables in hierarchy
- creation date
- description.



Display 1

This information can be very helpful when modeling MDDB's.

## Incremental Update

It is not unusual for customers to have gigabytes of source data that need to be summarized. Summarizing this volume of data can take hours. Because of this, it is important to be able to update the MDDB with new data without having to completely rebuild the MDDB. The SAS/MDDB Server allows incremental updates or *drip feeding*.

To update an existing MDDB, the data set containing the update data must be specified, as well as an IN= and an OUT= MDDB. It is important to remember the IN= and the OUT= MDDB must be different. You can also specify a label for the updated MDDB using the LABEL= option. When updating a MDDB, all CLASS, VAR, and HIERARCHY statements are ignored.

Using the previously created MDDB called SASUSER.TESTMDDB, you can incrementally add new values from the SAS data set SASHELP.PRDSAL2. The following code will do this:

```
proc mddb data=sashelp.prdsal2 in=sasuser.testmddb  
  out=work.testmddb label="updated 08Dec96";  
run;
```

You have now created a second MDDB called WORK.TESTMDDB. To move that over to the existing MDDB, use the PROC COPY command. An example of this follows:

```
proc copy inlib=work outlib=sasuser memtype=mddb;  
  select testmddb;  
run;
```

*continued on next page*

WARNING: From the syntax, it should be obvious that there will be two copies of the MDDB present during the update. Make sure there is enough disk space allocated to the target library before the update is attempted.

## SIZE Formula

A topic of great concern for those getting ready to build a MDDB is the size. To predict the size of an MDDB, you need to know the following:

- number of analysis variables
- number of class variables
- maximum formatted length of each class variable
- length of each class variable
- number of distinct values in each class variable.
- number of valid crossings between class variables for each table.

The formula for the size of a MDDB follows:

For every MDDB:  
900 bytes

For every Analysis Variable:  
676 bytes

For every Class Variable:  
640 bytes +(maximum formatted length\*number of values)+  
(length\*number of=values)

For each table/sub-table(always at least one)  
288 bytes + 8 \* number of class variables

For each sub-table:  
((number of dimensions\*sizeof(int))+  
(number of analysis vars\* number of stats\*sizeof(double))) \*  
number of valid crossings

## Cross Platform Access

While it is possible to build and exploit the MDDB on the same platform, it is quite likely that the MDDB will not be stored on the same platform as the client that will use it. MDDB's can be stored on servers, and then accessed via SAS/CONNECT or SAS/SHARE with SAS/EIS on clients. The Remote Library Services (RLS) function found in SAS/CONNECT and SAS/SHARE can make access to this data dynamic. Accessing remote data using RLS may be faster with Release 6.12 and the MDDB because of the read only nature of the product. After logging into a remote machine, a local Libref can be assigned to a path on the remote machine. The syntax is as follows:

```
LIBNAME libref REMOTE 'remote-pathname' SERVER=servername;
```

This should be executed on the local machine and will create a local reference to any remote MDDB (or SAS file) that exist on the remote machine in that path.

## Program Availability

The programs used in this article to create MDDBs (makemddb.sas) and to update MMDBs (dripmddb.sas) are available online through Anonymous FTP. To access this service, see the instructions inside the front cover of the journal.

## Conclusion

With the MDDB Server the SAS System can provide faster access to data for online analysis. PROC MDDB was the focal point for this discussion, but this information can easily be translated to SAS/EIS or SAS/AF to build the MDDB. ●

SAS, SAS/AF, SAS/CONNECT, SAS/EIS, SASA/MDDB, and SAS/SHARE are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.