The SAS® Multidimensional Database and Its Uses in Applications Development
Bill Powers, SAS Institute Inc., San Francisco, CA

ABSTRACT
The SAS multidimensional database (MDDB) provides an efficient and powerful data storage structure for summarized data. Data stored in the MDDB data structure can be used by standard SAS procedures and in applications developed with the SAS/EIS® and SAS/AF™ modules. This tutorial discusses when and how to create an MDDB both with programming statements and through the SAS/EIS point-and-click interface. Consideration will be given to efficiency issues and updating an existing MDDB. Examples will be presented demonstrating how to use an MDDB with SAS procedures, with selected SAS/EIS objects, and in SAS/AF FRAME entries.

INTRODUCTION
The SAS System MDDB is a read-only file format for the storage of summarized data. An MDDB is not a SAS data set and you cannot process it directly using SAS procedures. However, you can utilize an MDDB in SAS/AF and SAS/EIS application windows and you can extract and process subtables of an MDDB with the DATA step and SAS procedures using the SASSFIO engine.

You may consider creating an MDDB whenever you have a data set with several classification and analysis variables and your application needs fast access to statistics summarized by any or all combinations of the classification variables. Each MDDB contains a minimum of a single NWAY summary table and, optionally, one to many separate subtables. The NWAY table is analogous to the output records produced when the NWAY option is used with PROC SUMMARY. Similarly, each additional MDDB subtable is analogous to PROC SUMMARY output records that all have the same value of the _TYPE_ variable.

FEATURES OF AN MDDB
All MDDBs
- are stored in SAS data libraries
- have a member type of MDDB
- can be viewed with SAS utility windows
- have no limit on the number of subtables
- have no limit on the number of CLASS variables
- have no limit on the number of ANALYSIS variables
- may contain up to 8 different statistics
- can create 13 other statistics at run time
- contain only valid class crossings (no missing data)
- store all data as character data
- store data in formatted form when formats are used.

MDDB CREATION ISSUES
PRE-CREATION DECISIONS
Before you build an MDDB, you should have a good understanding of
- the data and how it will be used
- the business problem
- user expectations.

Building the correct MDDB requires a delicate balance between the need for fast reporting response times and the need to conserve storage space. Knowing your data and the business problem will enable you to determine how the data should be summarized, how views and formats can be used to create summarization levels that are not present in the raw data, and what statistics should be stored. Knowing your user’s expectations will assist you in determining which and how many subtables to create. Finally, building an MDDB may be an iterative process tailored to complement the overall data warehousing and OLAP strategies of your company.

SUBTABLE ISSUES
Many MDDBs are created to be used as data sources for SAS/EIS objects such as the Multidimensional Report and Business Graph objects. These and other SAS/EIS objects have support for runtime user modifications such as drill down, expanding, rotation, changing of drill hierarchies, swapping of dimensions, and selection of analysis variables. This flexibility, combined with user’s desire for fast response times, suggest that the MDDB contain every possible summarization (subtable). However, such a strategy may result in an MDDB that requires considerable CPU time to create and a prohibitive amount of storage space. Additionally, performance may actually suffer slightly because of the time required to determine which single subtable (from among hundreds of potential subtables) contains the combination of class variable values that closest meets the user’s request.

AN EXAMPLE
Consider a SAS data set with one analysis variable, SALES, and the character classification variables shown in Table 1.

Table 1 Classification Variables

<table>
<thead>
<tr>
<th>VARIABLE NAME</th>
<th>DISTINCT LEVELS</th>
<th>LENGTH</th>
<th>FMT LEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION</td>
<td>EAST, WEST</td>
<td>$ 1</td>
<td>$ 4</td>
</tr>
<tr>
<td>DISTRICT</td>
<td>1, 2, 3 *</td>
<td>$ 1</td>
<td>$ 10</td>
</tr>
<tr>
<td>TERR</td>
<td>A, B, C, D</td>
<td>$ 1</td>
<td>$ 10</td>
</tr>
<tr>
<td>YEAR</td>
<td>1996, 1997</td>
<td>$ 4</td>
<td>$ 4</td>
</tr>
<tr>
<td>QUARTER</td>
<td>1, 2, 3, 4</td>
<td>$ 1</td>
<td>$ 1</td>
</tr>
</tbody>
</table>

* EAST has two DISTRICTS 1, 2
WEST has three DISTRICTS 1, 2, 3

In a drill-down scenario you would likely define two drill hierarchies

** GEOGRAPHIC: REGION -> DISTRICT -> TERR **

** TIME: YEAR -> QUARTER. **

An application window might initially display the following table.

Table 2 Initial Display

<table>
<thead>
<tr>
<th>REGION</th>
<th>SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>4000</td>
</tr>
<tr>
<td>WEST</td>
<td>7000</td>
</tr>
</tbody>
</table>

After clicking to expand both the EAST and WEST rows, you see Table 3.
In order to provide the best drill-down response time to the user, you need two subtables in the MDDB - the REGION subtable for the initial display and the REGION"DISTRICT table for expanding (or drilling down). If the user expands or drills down further in Table 3, you need the REGION"DISTRICT"TERR subtable.

If you now add TIME as an across dimension to your application, the initially displayed table is shown below.

### Table 4 Initial Table with Across Dimension

<table>
<thead>
<tr>
<th>REGION</th>
<th>SALES 1996</th>
<th>SALES 1997</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAST</td>
<td>2500</td>
<td>1500</td>
</tr>
<tr>
<td>WEST</td>
<td>4000</td>
<td>3000</td>
</tr>
</tbody>
</table>

Table 4 is displaying the cells in the REGION"YEAR subtable. If the user is now allowed to drill down in the row dimension, you also need the REGION"DISTRICT"YEAR subtable and the REGION"DISTRICT"TERR"YEAR subtable. Carrying this example to its extreme by allowing the user to simultaneously drill down in the across dimension (TIME), you need REGION"YEAR"QUARTER, REGION"DISTRICT"YEAR"QUARTER, and REGION"DISTRICT"TERR"YEAR"QUARTER subtables. Note that this last subtable is, in fact, the NWAY table that is always generated in an MDDB.

If the required table corresponding to any expand or drill down request from the user is not available, the SAS software determines and uses the existing table that requires the least roll-up summarization. Given the small total number of cells in all of the above-mentioned subtables, very acceptable performance could likely be realized by creating the MDDB with only the default NWAY table and performing all necessary roll-up summaries at run time.

### STORED AND DERIVED STATISTICS

An MDDB can store a maximum of 8 different statistics. Depending on which of these 8 possible statistics are actually stored, 13 others can be derived at run time. The 8 statistics that can be stored in an MDDB are:

- N
- SUM
- SUMWGT
- UWSUM
- NMISS
- USS
- MIN
- MAX
- WEIGHT.

Notice that the MDDB procedure uses the same keywords for statistics as the SUMMARY and MEANS procedures.

Storing only simple statistics and valid class crossings helps keep the MDDB size down to a minimum thereby keeping MDDB build time short. The following table indicates which statistics you should store in an MDDB to have access to certain statistics at run time.

### Table 5 Necessary Versus Derived Statistics

<table>
<thead>
<tr>
<th>Wanted</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>SUM</td>
<td>SUM</td>
</tr>
<tr>
<td>SUMWGT</td>
<td>SUMWGT</td>
</tr>
<tr>
<td>UWSUM</td>
<td>UWSUM</td>
</tr>
<tr>
<td>NMISS</td>
<td>NMISS</td>
</tr>
<tr>
<td>USS</td>
<td>USS</td>
</tr>
<tr>
<td>MIN</td>
<td>MIN</td>
</tr>
<tr>
<td>MAX</td>
<td>MAX</td>
</tr>
<tr>
<td>MEAN</td>
<td>N,SUM</td>
</tr>
<tr>
<td>RANGE</td>
<td>MIN,MAX</td>
</tr>
<tr>
<td>PCTN</td>
<td>N</td>
</tr>
<tr>
<td>PCTSUM</td>
<td>SUM</td>
</tr>
<tr>
<td>CSS</td>
<td>N,SUM,USS</td>
</tr>
<tr>
<td>VAR,STD,STDERR,CV,T,PRT,L</td>
<td>N,SUM,USS</td>
</tr>
<tr>
<td>CLM, UCLM</td>
<td></td>
</tr>
</tbody>
</table>

### BUILDING THE MDDB WITH PROC MDDB

You can create an MDDB in a point-and-click environment using SAS/EIS or by submitting statements, either interactively or in batch jobs, with the MDDB procedure.

### THE MDDB PROCEDURE

The syntax for the MDDB procedure is:

```sas
proc mddb <data=dsname> out=lib,mddb
   <in=orig.mddb> <label=description>
   <class class_var_list </order-options> >;
   hierarchy class_var_list </
   <name="name"><display=YES|NO>>;
   var analysis_var_list </stat-options>
   run;
```

The syntax of the MDDB procedure is very similar to the syntax of the SUMMARY procedure. If you are already familiar with the SUMMARY procedure and the output data set it can produce then you already have a good understanding of the SAS system MDDB.

### EXAMPLES

**Example 1** Create an MDDB for the company sales data with only the NWAY table and accepting all defaults.

```sas
proc mddb data=prem.sales out=perm.salemddb;
   class region district terr year quarter;
   var sales;
run;
```

In this example, only the default NWAY table is created and the NWAY table has 160 cells.

**Example 2** Create an MDDB for the company sales data with the GEOGRAPHIC and TIME drill hierarchies in mind and specify customizing options.

```sas
proc mddb data=prem.sales out=perm.salesmddb;
   class region district terr year quarter;
   var sales;
run;
```
Two subtables are created in this example in addition to the NWAY table. Individual CLASS statements are used for each class variable to control the order in which the class values are displayed in reporting. Options on the VAR statement specify the statistics to compute and store for the analysis variable SALES. Note that the MEAN statistic will be available for reporting since it can be derived at run time from the SUM and N statistics. The HIERARCHY statements specify the subtables to create, the NAME= option enables you to provide a descriptive label for the hierarchy, and the DISPLAY= option determines if the hierarchy is automatically used by SAS/EIS as a drill hierarchy in the metabase registration process. If you do not name hierarchies, they are named HIER1, HIER2, ..., HIERN.

Example 3  Create an MDDB for the company sales data with the GEOGRAPHIC and TIME hierarchies and additional subtables for optimal drill-down performance when each hierarchy is used separately or they are used together (one down, the other across).

```
proc mddb data=prem.sales out=perm.salemddb;
  class region/desorder;
  class district/ascformatted;
  class terr/desformatted;
  class year/descending;
  class quarter/ascending;
  var sales/sum n min max;
  hierarchy region district terr/
    name='Geographic Dim' display=yes;
  hierarchy year quarter/ name='Time Dim'
    display=yes;
run;
```

ESTIMATING THE SIZE OF AN MDDB

The following information enables you to estimate the size of an MDDB based on the number of analysis variables, statistics, subtables, and levels of class variables.

```
For each MDDB:
  900 bytes overhead
For each analysis variable:
  676 bytes overhead
For each class variable:
  340 bytes overhead +
    (max_formatted_length*num_levels)+
    (unformatted_length*num_levels)
For each subtables:
  296 bytes overhead +
    (num_dimensions*4 +
    num_analysis_vars*num_stats*8)*
    num_of_crossings_in_subtable
```

Example 4  Estimate the size of the MDDB created for the SALES data set shown in Table 1 with the statements shown in Example 2.

<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Subtable</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION</td>
<td>NWAY</td>
<td>350</td>
</tr>
<tr>
<td>DISTRICT</td>
<td>NWAY</td>
<td>373</td>
</tr>
<tr>
<td>YEAR</td>
<td>NWAY</td>
<td>356</td>
</tr>
<tr>
<td>QUARTER</td>
<td>NWAY</td>
<td>344</td>
</tr>
<tr>
<td>GEOGRAPHIC</td>
<td>NWAY</td>
<td>1176</td>
</tr>
</tbody>
</table>

The actual size of the MDDB build with the statements in Example 2 is 16640.

INCREMENTALLY UPDATING AN MDDB

You may have gigabytes of source data that needs to be summarized and stored in an MDDB. This process may take considerable CPU time. As additional source data is collected over time, you need to incorporate this new data in the MDDB without completely rebuilding the MDDB. The syntax of the MDDB procedure supports this incremental updating (or "drip feeding") of an existing MDDB.

Example 5  Incrementally updating the existing MDDB, PERM.SALEMDDB with the new data stored in the SAS data set PERM.NEWSALES.

```
proc mddb data=prem.newsales in=perm.salemddb
  out=work.newmddb;
run;
```

Once the step has completed and you have verified it executed successfully, you can use the DATASETS procedure to delete the old MDDB, PERM.SALEMDDB, and then copy WORK.NEWMDDB to PERM.SALEMDDB.

BUILDING THE MDDB WITH SAS/EIS

You can also use the point-and-click SAS/EIS environment to create an object which, when executed with the RUNEIS command, will create an MDDB. The RUNEIS command can be executed interactively or in a batch job. In addition to the benefits of a point-and-click environment, the SAS/EIS approach

- uses information about the source data set that is already stored in the SAS/EIS metabase
- uses information about the drill-down hierarchies already registered in the SAS/EIS metabase
- automatically registers the MDDB in the SAS/EIS metabase so that it can readily be used as a data source for selected SAS/EIS objects
provides access to most of the options available in the MOOB procedure

- offers three criteria for determining which subtables to create and then automatically generates the selected subtables.

The three criteria offered in the SAS/EIS object are

- most paths covered - a compromise between space and speed (default)
- use minimal disk - smallest number of subtables (NWAY table only) and minimum amount of disk space
- best performance - largest number of subtables and largest amount of disk space.

USING AN MDDB

USING AN MDDB IN SAS/EIS APPLICATIONS

The SAS/EIS software offers numerous predefined object for analysis and reporting. Many of these object enable the user to explore the data by drilling down, subsetting, changing hierarchies, rotating dimensions, and exchanging analysis variables at run time. Such features are obviously most effective when response time is fast. This necessitates working with a summarized data source.

In Release 6.12 of the SAS/EIS software, the following objects can utilize an MDDB as the data source:

- Multidimensional report
- 3D Business graph
- Map
- Organizational chart
- Graphical variance report

No special effort is required to utilize an MDDB with these objects other than registering the MDDB in the SAS/EIS metadata. Then the MDDB is simply selected as the source table for the object in the object's attribute window at build time.

The objects above can also accept SAS data sets as their data source. However, if these objects are pointed to a SAS data set, a temporary MDDB will automatically be built at run time.

USING AN MDDB IN SAS/AF APPLICATIONS

Numerous methods are supplied with SAS/EIS for accessing the summarized data stored in an MDDB as well as information about the class and analysis variables. You can call these methods using Screen Control Language (SCL) while developing SAS/AF FRAME applications or in overriding the default behavior of SAS/EIS objects.

Example 6 Create a SAS/AF FRAME to enable the user to select any subtable of an MDDB from a list box and then display the subtable in a data table object. For performance considerations, the data table object should display the smallest subtable when the FRAME window is first displayed to the user.

First, create the FRAME window with two objects - a list box object named LISTBOX and a data table object named TABLE. Populate the list box object from an SCL list named NAMELIST.

Second, write the SCL code to perform the following tasks:

1. create an MDDB object and point it to the SALES MDDB
2. determine the names and sizes of all subtables (including the NWAY table)
3. populate the list box object with a list containing the names of all the subtables
4. identify the subtable with the fewest number of cells
5. initialize the data table object with that subtable
6. determine the subtable selected when the user clicks in the list box object
7. repopulate the data table object with the user selected subtable
8. terminate the MDDB object and delete all lists when the user closes the FRAME window.

The necessary SCL to perform steps 1-8 is shown below.

```scl
length curname $40
tabexist $1;

INIT:
    /* Load the model and attach to TABLE */
    classid=loadclass('SASHELP.EIS.MDBB_M.CLASS');
    mddbid=instance(classid);
    call notify('table', '_ATTACH_ ', mddbid);
    /* Set the MDDB to the SALES MDDB */
    call send(mddbid, '_SET_MDBB_TABLE_ ',
             'perm.sales');
    /* Get list of hierarchy info from MDDB */
    sublist=makelist();
    call send(mddbid,'_GET_HIERARCHIES_' ,
             sublist);
    /* Get list of hierarchy names and sizes */
    hlist=getniteml(sublist,'HIERARCHY');
    ncell=getniteml(sublist,'NUMBER_OF_CELLS');
    /* Find smallest subtable to display first */
    mincells=getitemn(ncell, 1);
    namelist=makelist();
    do i=1 to listlen(hlist);
        curcells=getitemn(ncell, i);
        mincells=min(mincells,curcells);
        namelist=insertc(namelist, nameitem(hlist, i), -1);
    end;
    minndx=searchn(ncell, mincells);
    /* Get name of smallest subtable */
    curname=getitemc(namelist, minndx);
    curlist=getniteml(hlist, curname);
    link DOTABLE;
    return;

DOTABLE:
    /* Display the currently requested subtable */
    call send(mddbid, '_TABLE_EXIST_ ',
              'CURTABLE', tabexist);
    if tabexist='Y' then
        call send(mddbid, '_DELETE_TABLE_ ',
                  'CURTABLE');
    call send(mddbid, '_MAKE_TABLE_ ',
              curlist, 'CURTABLE');
    return;

LISTBOX:
    /* Determine selected table */
    call notify('LISTBOX', '_GET_LAST_SEL_',
                row, issel, curname);
    curlist=getniteml(hlist, curname);
    link DOTABLE;
    return;
```
TERM:
    call send(mddbid,'_TERM_');
    if sublist then sublist=dellist(sublist,'Y');
    if namelist then 
        namelist=dellist(namelist,'Y');
return;

USING MDDB SUBTABLES WITH SAS PROCEDURES

The SAS data engine SASSFO can be used to surface subtables in an MDDB so that they can be processed directly with SAS procedures.

The syntax to access MDDB subtables is:

    libname libref SASSFO 'mdbb_physical_file_name';
    proc procedure_name DATA=libref.subtable;
    run;

Example 7 Use the PRINT procedure to produce a report from the NWAY subtable in the SALES MDDB.

    libname in sassfio 'd:\salesdata\salesmdb.md2';
    proc print data=in.nway;
    run;

Note that under desktop operating systems a SAS MDDB has an extension of MD2.

CONCLUSIONS

The SAS Multidimensional database is a data storage structure for the efficient storage and retrieval of summarized data that is particularly suited for applications development projects that support drill-down analyses with multiple drill hierarchies. You can create an MDDB in both batch and interactive environments and, as new source data is added, you can incrementally update an existing MDDB. An MDDB is easily accessed with SAS/EIS objects, SAS/AF FRAMEs, and SAS procedures.

ACKNOWLEDGEMENTS

Discussions with Fritz Lehman and David Home of SAS Institute Inc., Cary, provided the foundation for this paper. The author also wishes to thank Tracy Folles and Regina Louie of SAS Institute Inc., San Francisco, CA for their suggestions and assistance with the final paper.

SAS, SAS/AF, and SAS/EIS are trademarks or registered trademarks of SAS Institute, Inc. in the USA and other countries. © indicates USA registrations.