Paper 139

The Art of Designing HOLAP Databases

Mark Moorman, SAS Institute Inc., Cary NC

ABSTRACT

While OLAP applications offer users fast access to information across business dimensions, it can also create very large databases. With the introduction of the HOLAP solution users can now design databases using multi-dimensional databases, SAS datasets, and relational data stores. This offers a lot of flexibility, but how are these databases designed to make the best use of each technology?

INTRODUCTION

This paper will explore how the HOLAP solution can be used to build large, remote databases that are specifically designed for navigation of data. It will explore how and when to use each type of technology so that the final outcome will be the best it can be. Finally, it will give some basic examples of building these multi-dimensional databases.

WHAT IS HOLAP?

To really answer this question, one must first understand what OLAP is. OLAP (OnLine Analytical Processing) applications are meant to allow users the freedom to interrogate their data. OLAP basically allows users to compare similar data points at any level of category. Basically, OLAP is supposed to offer fast access to data along any business dimension. To allow this, technology companies took two distinctly different paths. Some introduced new databases built specifically for multi-dimensional queries, while others offered very sophisticated SQL and navigational methods on top of traditional RDBMS databases. As is often the case with technology companies they decided to use this different data store as a competitive difference in the marketplace. This competition created the MOLAP (Multidimensional OLAP) and ROLAP (Relational OLAP) camps. In reality both had very useful qualities for OLAP applications. So it is that another alternative came to be. Basically, HOLAP (Hybrid OLAP) is the bridging of this technology gap. HOLAP allows the use of both MDDB and RDBMS data stores for the best of both worlds.

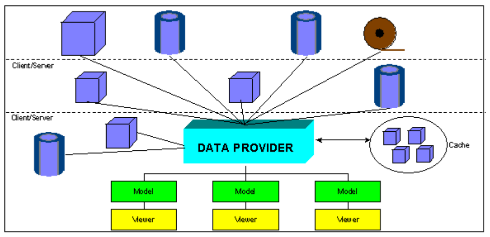


Figure 1.1 HOLAP data Model. The Data Provider navigates the virtual cube, sends off the proper SQL or MDDB request, and then forwards the result on to the Viewer.

WHEN DO YOU MOLAP, ROLAP OR HOLAP?

This is not as simple a question as it may seem. The first and most elementary answer might be to use MOLAP until it no longer works, and then use HOLAP. This has some truth to it. Basically MOLAP databases (all Multi-dimensional in nature) are faster, and easier to maintain. MOLAP databases are one file on the filing system. The navigation is inherent, and not implied. The indexes are automatically built and designed for the best response when doing OLAP queries. The limitation of MOLAP is that it is not as scalable as ROLAP and HOLAP. While the MDDB data store is getting better, there is still a limitation to the amount of data that it can hold. In version 6.12 of the SAS system, that amount is 2 gig for any Specific Crossing. A Specific Crossing (sometimes referred to as a sub-table) is a section of the MDDB that holds a specific crossing of data. The dimensions chosen create crossings of data. So if the dimensions of an MDDB are Time, Geography, Product and Scenario, then a specific crossing might be Year by Quarter by State. So, the technical limit might be 2 gig for a Specific Crossing, but managing files this large often require the controls in databases like SPDS and other RDBMS. So it may be that users need to switch to HOLAP before they reach the technical limits.

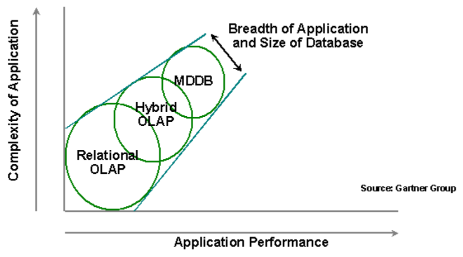


Figure 1.2 Chart of OLAP Types from the Gartner Group. They define the relationship of application performance, and complexity of application to the ROLAP/MOLAP/HOLAP environments.

WHAT IF THE DATA WAREHOUSE IS ALREADY BUILT?

Many organizations have already built data warehouses using a data model called Star Schema. This is a relational data model using join technology to equate the dimensions. It is very popular in organizations that are building corporate wide data warehouses. The Star Schema is basically a fact table that represents the actual values present in the database, and any number of dimension tables. Dimension tables reflect business dimensions like Time and Geography. So to find the proper data a query selects a join of the appropriate dimension table, and the fact table based on that dimensions key. This will quickly retrieve all of the data relating to that level of the fact. This is a good way to build a large central data warehouse for generalized use. However, it is not very easy to build or manage. In this model the keys, the relationships, and the navigation are all managed by the RDBMS. This creates a pretty large overhead. If the purpose is quick ad-hoc reporting on a specific set of data, it is better to build a datamart using MDDB technology. However, if all of the work has already been done, or is being done by another group in an organization then the SAS HOLAP solution can tap into that data repository without having to replicate the data.

QUICK COMPARISION ROLAP AND MOLAP

While none of these statements are hard fast rules, and there can always be an exception found, these basic differences generally exist.

MOLAP

- Fast response time
- Easier maintenance
- Less scalable

ROLAP

- Extremely scalable
- Uses current technologies

- Generally slower
- More maintenance and design

In SAS you can use both a MOLAP and/or ROLAP approach. The ROLAP approach can be implemented using the SAS HOLAP extensions.

HOLAP THE BEST OF BOTH WORLDS

So with HOLAP, either implementation or mix can be created. The question then becomes what are implementation guidelines. As with the MOLAP and ROLAP question, there is really no “right” answer although there are some very reasonable guidelines to follow. If it is true that MOLAP should be used until it can no longer scale, then how do you break the database after that? Is it necessary to use ROLAP if the model is too big for MOLAP? The best results can be found by examining the data. To really design an OLAP database one must understand the dimensionality of the data.

WHAT IS IT ABOUT OLAP DATA?

It is common practice today to see data as a collection of rows and columns, but that doesn’t really work in OLAP. The question is not can the OLAP database handle one million rows of data, but can it handle the relationships that exist in the data. One million rows of data means nothing in OLAP terms. It is quite possible to store one million rows of data in two cells. For instance if the relationship in the data was SEX, then there would only be two cells; one with the total of all males, and another with the total of all females. So the number of rows has a smaller impact on the actual size of an OLAP database than the crossings of data. The real factors in an OLAP database are

- Number of categories
- Number of unique values for each category
- Number of statistics stored for each category
- Number of specific crossings (sub-tables)
- Amount of sparsity

The problem with this model is often the number of unique values in a category is unknown, and even more often, the sparsity of the data is unknown. Sparsity has a big impact on the amount of storage needed for a SAS/MDDB® database. Many OLAP databases store sparsity, and therefore it has no impact on the size for those databases. For those vendors the size of the actual database could be very large, because it stores a cell at every crossing of category variables. SAS/MDDB® does not store any empty cells, and so to really get a feel for how big the database might be, the size has to be reduced by the amount of sparsity. This makes for smaller databases, but harder calculations. One way to get a feel for this information is to run a PROC FREQ. This will tell how many values each level of crossing has. By putting the output in a dataset, it can be used to predict size of Specific Crossings, and the total size of the database.

```
proc freq data=SASUSER.PRDSALE  ;
   tables COUNTRY*REGION*DIVISION*PRODTYPE /
noprint out=WORK.TEST
   missing missprint cellchi2 ;
run;
```

In most applications, the data has only a few categories that make up the bulk of the size. For example categories for MONTH and YEAR are generally small. Usually MONTH has only 12 unique values while, categories such as CUSTOMER and SKU (product) can be very large having perhaps millions of unique values. These categories with high cardinality (number of unique values) create the bulk of the data size. When they are crossed with MONTH and YEAR they create the problem.

AN OLAP DATA MODEL

Once again there is no right way to build an OLAP database. The best method would be to know the exact queries that were going to be run against it, and build just those crossing. However, the real strength of OLAP is to compare things that may have been hidden for years. To do that an ad-hoc environment has to be designed. Short of actually designing the database around the exact queries, there is a way to build a good OLAP data model for ad-hoc queries. OLAP databases are designed to compare similar values across any number of business drivers. The question may be how is the sale of hats in Florida vs. the sale of hats in Georgia over the last six months in the superstores. This query uses six dimensions. The dimensions are

Sales
Products (hats)
Geography (Florida)
Time (months)
Channel (superstores)

This query is quite normal, and uses only one level of six different dimensions. However, many people build Specific Crossings by dimension (YEAR, MONTH, DAY). Since many queries include several dimensions a better way might be a Spiral approach.

The Spiral design takes each of the dimensions, and places them on an axis starting with the highest level, which often happens to be the lowest cardinality (least unique values), in to the lowest level of the dimension which is often the highest cardinality. Then to build the crossings, start at the axis of the graph, and work out. Each layer creates another specific crossing.

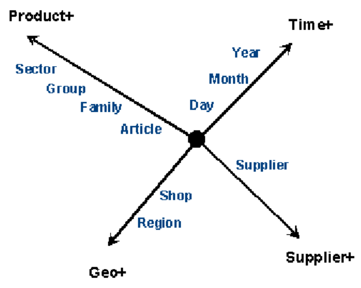


Figure 2.1 Spiral HOLAP data model: Starting with the highest cardinality and working out along an axis for each dimension.

Using the example in Figure 2.1, the ordered list of classes is:

YEAR
SECTOR
REGION
GRP_SUPP
MONTH
GRP
SHOP
SUPPLIER
FAMILY
DAY
ARTICLE

If this order is reversed it becomes the prioritized class list. To produce the choice of crossings, copy the previous line and drop the first item for example:

```
article day family supplier shop grp month
grp_supp region sector year

day family supplier shop grp month grp_supp
region sector year

family supplier shop grp month grp_supp
region sector year

supplier shop grp month grp_supp region
```

```
sector year

shop grp month grp_supp region sector year
grp month grp_supp region sector year

month grp_supp region sector year

grp_supp region sector year

region sector year

sector year

year
```

If the data is being dominated by one or two large categories, then it is a candidate for a one on one data model. The one on one data model basically takes the large categories, and creates a fact table from them in a SAS or RDBMS file. It then takes all of the other categories, and stores them using the MDDB. Then a simple HOLAP structure is completed linking the table and the MDDB. This model uses the strength of both the MDDB and RDBMS data stores. Since the MDDB's real strength is navigating lots of similar crossings to find the best one, it is best when storing many small to medium size Specific Crossings. Since RDBMS data stores are better at finding a record or small group of records in a much larger group, it is best used when storing large specific crossings. Using this model also creates a new level of openness. Since the relational store is basically the lowest level of summarization, and is stored in a SQL format, it is now open to non-OLAP queries. This level of summarization may already exists in the organization today. It is important to note that when using the one on one model, that more Specific Crossings are stored in the MDDB. The RDBMS should not be used for queries that require rollup, and only queries that are selecting a subset of the overall crossing. It is also a good idea to index the RDBMS file on the high cardinality categories since these categories will most often be used for reporting.

WHAT IS DYNAMIC ROLLUP

When using the HOLAP solutions, the Data Provider makes a distinction about which Specific Crossing to access. The Data Provider basically tries to choose the smallest set of data that will answer the question. This may actually mean that no Specific Crossing is an exact match for the question. When that is true the HOLAP engine rolls up the next largest Specific Crossing which has the appropriate data in it. This means that for the query YEAR by PRODTYPE, that the Data Provider may have to choose the Specific Crossing of YEAR by QTR by MONTH by PRODTYPE, because that is the next smallest Specific Crossing that has YEAR and PRODTYPE in it. When using the one on one model it is possible to create rather small queries that read the large RDBMS Specific Crossing. Using the Spiral technique in conjunction with the one on one model usually protects against this.

ANOTHER DESIGN

You can also use a bitmap to represent the crossings. A 1, indicates absence a 0, indicates presence of a class. So the previous representation becomes:

Bitmap	Decimal
1 1 1 1 1 1 1 1 1 1 1	2047
0 1 1 1 1 1 1 1 1 1 1	1023
0 0 1 1 1 1 1 1 1 1 1	511
0 0 0 1 1 1 1 1 1 1 1	255
0 0 0 0 1 1 1 1 1 1 1	127
0 0 0 0 0 1 1 1 1 1 1	63
0 0 0 0 0 0 1 1 1 1 1	31
0 0 0 0 0 0 0 1 1 1 1	15
0 0 0 0 0 0 0 0 1 1 1	7
0 0 0 0 0 0 0 0 0 1 1	3
0 0 0 0 0 0 0 0 0 0 1	1

If you look closely you see that these crossings can be derived from:

$2^n - 1$ where n is the number of crossings

This means these crossings can be used for runtime summarization. The key is getting the order of the crossings correct. The knowledge of hierarchies and cardinality provide the best starting point.

Sometimes it is also useful to store the lowest 7, 15 or even 31 bitmap values as these are typically small (the high cardinality is on the left). For example:

Bitmap	Decimal
0 0 0 0 0 0 0 0 1 1 1	7
0 0 0 0 0 0 0 0 1 1 0	6
0 0 0 0 0 0 0 0 1 0 1	5
0 0 0 0 0 0 0 0 1 0 0	4
0 0 0 0 0 0 0 0 0 1 1	3
0 0 0 0 0 0 0 0 0 1 0	2
0 0 0 0 0 0 0 0 0 0 1	1

In our example this might be:

REGION	SECTOR	YEAR
REGION	SECTOR	_
REGION	_	YEAR
REGION	_	_
_	SECTOR	YEAR
_	SECTOR	_
_	_	YEAR

Overall the aim is to find a balance between providing the best possible response times and storing as small a number of crossings as possible to reduce storage. The ability to perform compute server processing at runtime makes this process much easier. The number of crossings stored can be quite small provided the most useful ones are included.

WHEN ITS NOT THAT SIMPLE

When the data is not being dominated by one or two high cardinality categories, what alternatives exist? Sometimes the number of categories and not the size of categories create the problem. When this happens it may take a little more thought as to how best approach the situation. One important thing to remember is that the HOLAP solution requires at least one Specific Crossing to have all of the categories in it. If all of the categories are not in at least one Specific Crossing then it may be that some request will not be met. Therefore, when there are many small categories that are to be stored it may be best to build one crossing of all crossings in an RDBMS file. This will create a bottleneck on some queries, but it will at least answer any question. After a Specific Crossing is created with all of the categories, it is best to build several MDDB's using as many of the categories as possible. In this model it is best to have some sense of what queries might be run, because it will be important to group categories that are likely to be used together into Specific Crossings. When using this method, it is best to us the log file and use and iterative process for building the model.

HOLAP LOGGING

One of the best by products of the HOLAP solution is logging. With version 6.12 a log file is built that contains most if not all of the information needed to tune a HOLAP database. From the log it can be determined how long the query took, how many cells it had to read, how many cells it should have taken, and the exact table or MDDB that was used to respond to the query. By using these logs it is easy to determine which queries are being run the most, and where the bottlenecks are. Using these logs, you can best match the needs of the customer to the limitations of the hardware.

The Log file includes the following information in Version 6.12.

Access Method
Access Time
Analysis
Cache MDDB Name
Cell Threshold
Classes
Datetime
DBMS Name
Exact Match ?
Member Type
Method
Object Name
Proposed Cells
Proposed Classes
Proposed Data Hierarchy
Proposed Data Name
Search Priority
Server Name
Subsets
SAS Version
User Identification

REMOTE ACCESS BENEFITS
Even when the amount of data does not require a HOLAP solution, using the HOLAP extensions may still be helpful. If the MDDB is on a server, and clients are accessing it via SAS/Connect®, then the HOLAP extensions will make it possible to use the server for computing. When in client server mode without the HOLAP extensions, all processing occurs on the client. Basically, the remote access selects the best Specific Crossing to use for the query then downloads it. If there is a need for rollup or sub-setting, then that takes place on the client. This means that more information was sent across the network than was needed. This can be a problem particularly in environments that have slow bandwidth on the network, and when cubes have extremely large Specific Crossings. With the HOLAP extensions, then only the exact data that matches the query is sent over the wire. This greatly reduces the amount of information flowing over the network, and eliminates a possible bottleneck.

SHOULD HOLAP EXTENSIONS ALWAYS BE USED IN C/S?
While the HOLAP extensions offer quite a bit of scalability, it is not true that they should be used whenever in a client server mode. It is possible by building many Specific Crossings to limit the network impact. Also, when the Specific Crossings are very small, it may not be needed. The HOLAP extensions can add more computing to the server, and this may in some small cases become the bottleneck. When using SAS/MDDB™ in client server mode it is always a good idea to determine the impact on the network and the server.

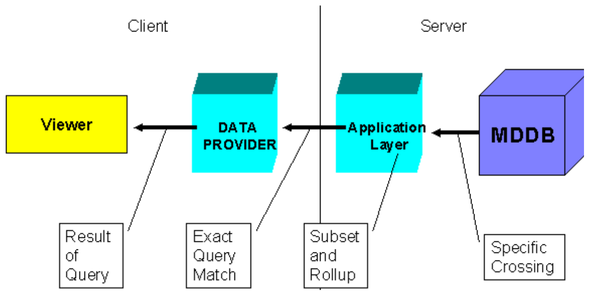


Figure 4.1 Client Server Model for the HOLAP extensions. Application logic submitted to the Server to cut down on Network traffic.

SO HOW DO YOU BUILD A HOLAP DATA STRUCTURE?
The steps to build a HOLAP data structure once the model is defined and the data has been accessed and cleansed are rather simple. Build each piece of the HOLAP structure using the proper technologies. Use PROC MDDB or SAS/EIS™ to build the MDDBs, and use the proper tool to build the tables that will be used.

NOTE: The Data Warehouse Administrator (SAS/WA®) has an extension available that can help with the building of the respective MDDBs and tables. While it is out of the scope of this paper, it can be very useful when building HOLAP data structures. More information can be found concerning the HOLAP extensions to SAS/WA® on the SAS Web site at www.sas.com.

Once the data sources that will make up the HOLAP structure are built, they can be linked together in the SAS/EIS® Metabase. There are two new attributes with the HOLAP extensions that can be used to define the HOLAP database. They are

- HOLAP Proxy MDDB (_DUMMDATA)
- HOLAP Associated data (_ASSDATA)

With these two properties the basic HOLAP solution can be defined.

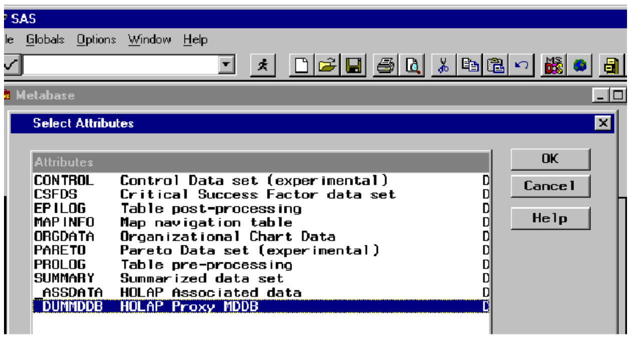
PROXY MDDB
The Proxy MDDB is used to store metadata locally on the client. This will speed up performance particularly at initialization time. It is not required, but should be used for best performance. The Proxy MDDB points to an MDDB structure that has all of the Metadata information, but no real data. Use a PROC MDDB to build a Proxy MDDB. The PROC should have all of the CLASS, VAR and DISPLAY HIERARCHIES that will be used in the HOLAP structure. It should use a dataset with one observation that has the full structure of the resulting HOLAP database. A sample PROC MDDB to build the Proxy may look like the following.

```
proc mddb data=data.nway (obs=1)
out=data.proxy ;

  class article day family supplier shop grp
month grp_supp region sector year;
  var qty sales / sum ;

  hierarchy year month day / name='Time+'
display=yes ;
  hierarchy grp_supp supplier /
name='Supplier+' display=yes ;
  hierarchy region shop / name='Geo+'
display=yes ;
  hierarchy sector grp family article /
name='Product+' display=yes ;
  RUN ;
```

Once the Proxy is built, it needs to be distributed to each client. This is a maintenance issue, but it will make for the best all around performance. It is a particular performance boost when the structure of the HOLAP database is complex and/or distributed. After the Proxy MDDB is built and distributed, it needs to be registered in the Metadata. To register the Proxy MDDB use the Metabase registration in SAS/EIS®. Add the new Proxy MDDB to the Metabase, and select the _DUMMDATA attribute. This is all that needs to be done to register the Proxy MDDB.



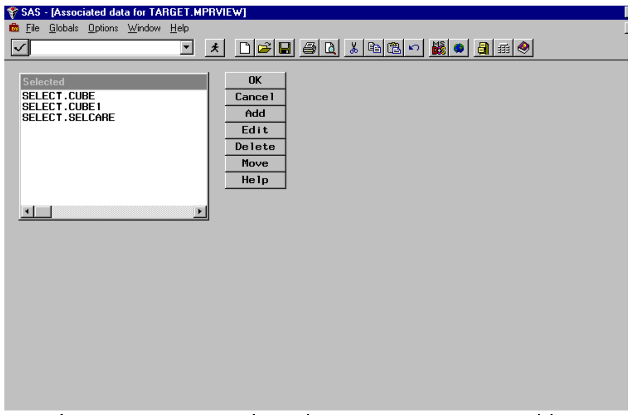
```
CONTROL    Control Data set (experimental)
CSFDS      Critical Success Factor data set
EPILOG     Table post-processing
MAPINFO    Map navigation table
ORGDATA    Organizational Chart Data
PARETO     Pareto Data set (experimental)
PROLOG     Table pre-processing
SUMMARY    Summarized data set
_ASSDATA   HOLAP Associated data
_DUMMDDB   HOLAP Proxy MDDB
```

Figure 5.1 Metabase registration for _DUMMDATA

HOLAP ASSOCIATED DATA

The Associated Data attribute is the heart of the HOLAP structure. It defines all of the pieces of information that make up the HOLAP structure. The Associated Data attribute is used to link each of the separate data items into one logical HOLAP database. With each piece of the HOLAP structure is stored information such as what data type it is, what server it is on, and relevant RDBMS information. To add an Associated Data attribute use SAS/EIS® Metabase to add to the Proxy MDDB the _ASSDATA attribute. When the Associated Data attribute is added, then any number of data sources can be added into the Associate Data grouping. Each data element in the HOLAP structure should be available via SAS/Connect® software using RLS at build time, but not at run time.

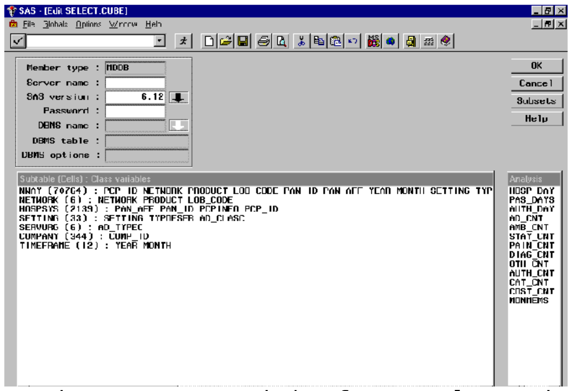
NOTE: When not using a Proxy, then SAS/Connect® can be used to create an RLS access to the data source that best represents the NWAY. That is the Specific Crossing that contains the greatest level of detail about the HOLAP organization.


SAS - [Associated data for TARGET.MPRVIEW]
File Globals Options Window Help
Selected
OK
Cancel
Add
Edit
Delete
Move
Help

```
SELECT.CUBE
SELECT.CUBE1
SELECT.SELCARE
```

Figure 5.2 Associated Data screen to add multiple data sources to one virtual MDDB

Once each of the separate data sources has been defined, specific information about each element can be defined via the EDIT button. From the EDIT Window, much of the information about this piece of the HOLAP structure is defined.


SAS - [Edit SELECT.CUBE]
File Globals Options Window Help
OK
Cancel
Subsets
Help
Subtable (Cells) : Class variables
Analysis

```
Member type : MDDB
Server name :
SAS version : 6.12
Password :
DBMS name :
DBMS table :
DBMS options :
```

```
NWAY (70764) : PCP ID NETWORK PRODUCT LOB CODE PAN ID PAN AFF YEAR MONTH SETTING TYP
NETWORK (6) : NETWORK PRODUCT LOB_CODE
HOSPSYS (2139) : PAN_AFF PAN_ID PCPINFO PCP_ID
SETTING (33) : SETTING TYPOFSER AD_CLASC
SERVURG (6) : AD_TYPEC
COMPANY (344) : COMP_ID
TIMEFRAME (12) : YEAR MONTH
```

```
HOSP_DAY
PAS_DAYS
AUTH_DAY
AD_CNT
AMB_CNT
STAY_CNT
PAIN_CNT
DIAG_CNT
OTH_CNT
AUTH_CNT
CAT_CNT
COST_CNT
MONMEMS
```

Figure 5.3 EDIT Window for one element in the _ASSDATA attribute.

After each element in the HOLAP structure has been properly defined, then the solution is ready to use.

USING HOLAP IN AN APPLICATION

Once the Proxy and Associated values are defined, the Metadata is complete. To actually use the HOLAP extension in version 6.12 the MODEL of the SAS/EIS® object must be overridden to use the new HOLAP model instead of the current MDDB model. Model overrides to SAS/EIS® objects are done in the Advanced portion of the Build process. In the Advanced portion of the Object properties, the MODEL should be changed to the HOLAP model. The HOLAP structure should then be in place.

CONCLUSION

The HOLAP extensions offer a great deal of flexibility when developing OLAP applications. HOLAP allows the integration of both Multi-dimensional and relational data stores to be used in concert for the fastest most scalable OLAP database available. Using the two together may present some unusual challenges, but the opportunities are enormous. The basic rules to remember are; use HOLAP when the amount of data gets too large for MOLAP; use HOLAP to help open bottlenecks in network throughput when using client server access; use the spiral design method to help visualize the cardinality of your data. Finally, HOLAP offers a truly scalable platform for enterprise wide OLAP applications.

REFERENCES

Much of the detailed material here can be found in SAS/EIS® Technical Report: HOLAP Extensions, Release 6.12.

ACKNOWLEDGMENTS

Thanks to Philip Mugglestone for the vision to come up with all of this. Thanks to Dave Horne and Fritz Lehman for technical guidance. Many thanks to Duane Ressler for building a great MDDB database.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mark Moorman
SAS Institute Inc.
SAS Campus Drive
CARY NC 27513
Work Phone: 919.677.8000 x6522
Email: sasmwm@unx.sas.com