Paper 319-2009

Best Practice: Optimizing the cube build process in SAS® 9.2

Mary Simmons, SAS Institute, Cary, NC
Michelle Wilkie, SAS Institute, Cary, NC

ABSTRACT

The loading and processing of your data into a cube needs to be controlled and optimized to get the best performance out of your system and meet appropriate time window constraints that may be set by IT or the business user. The SAS® OLAP Server 9.2 shows several performance improvements and changes in the cube building process compared to that in SAS® 9.1.3. This paper will introduce factors that affect the cube building process; options that are available to control and optimize the cube building process; best practices for cube building; logging options and understanding possible problems that can occur, in relation to these improvements in SAS® 9.2.

INTRODUCTION

SAS OLAP Server (On-Line Analytical Processing) provides users with the ability to build and view cubes. Cubes allow users to gain knowledge of the data by slicing, dicing and analyzing it in a summarized, multidimensional approach. This helps surface interactions within the data so that users can make informed decisions about their business. It is essential for these decisions, therefore, to make sure that the data is served to the user in a timely and relevant fashion.

The SAS OLAP Server supports Multidimensional OLAP (MOLAP), Relational OLAP (ROLAP) and Hybrid OLAP (HOLAP) structured cubes. A SAS MOLAP structure is a proprietary data store, in which data is feed from a warehouse into the cube building process; for summarization, defining data relationships via metadata, and computation of analytical variables. To build such a cube is simple; however, it is important to understand the cube building process so that:

- the appropriate SAS System options and PROC OLAP options are utilized correctly
- the system utilization of memory, I/O and storage is optimized
- the cube can be built successfully and in the time window necessary

Getting the OLAP cube into the business users' hands as fast as possible is the ultimate goal of optimizing the cube build process. The major focus of this paper will be on changes made in the SAS 9.2 cube build process; this will be illustrated using PROC OLAP.

SAS 9.2 PROC OLAP

In SAS® 9.2, the underlying OLAP cube structure changed significantly, automatically providing several benefits including:

- Better performance for the cube building process
- Lifting the Hierarchy member limitation
- Significantly faster member metadata creation
- Updateable cubes (providing incremental updates)
- Better query performance allowing faster response times for reporting applications surfacing OLAP data

In addition to these benefits, it may be possible to further optimize your cube build process. To do so, it is important to understand the underlying components of the cube, the phases of the PROC OLAP process, the options that apply to each phase *and* to understand the needs of your environment so that you can make the most of the resources that are available to you.

Cube Components and Build Phases

In Figure 1, you will see the OLAP cube broken down into its components. Cubes contain a Metadata Registration, navigational files that contain internal cube member metadata (separate from the Metadata Registration), as well as the NWAY and aggregation tables and indices.

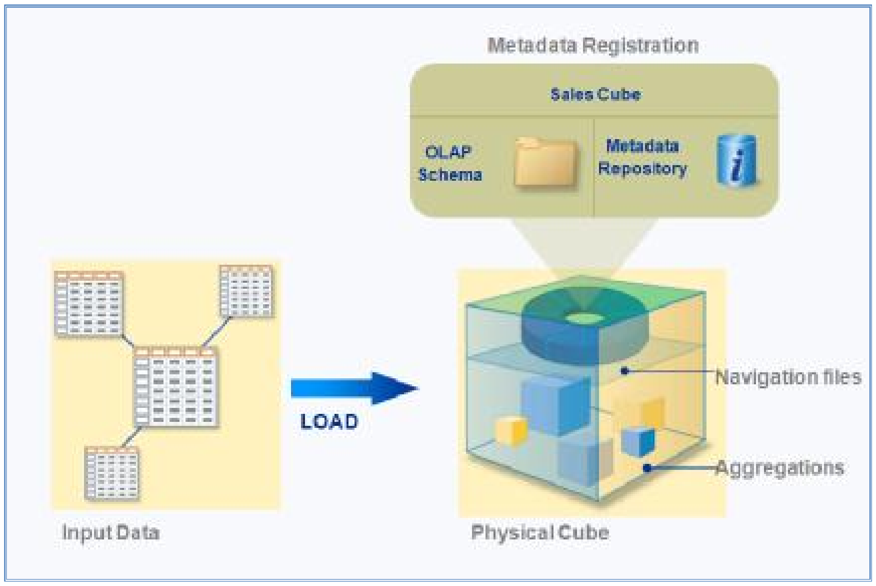


Figure 1 Components of an OLAP cube

The cube build process can be broken down into 2 major phases that are closely related to the cube components. Phase I is when the data summarization begins and the navigation files are created. In Phase II the NWAY, Aggregation tables and related indices are built.

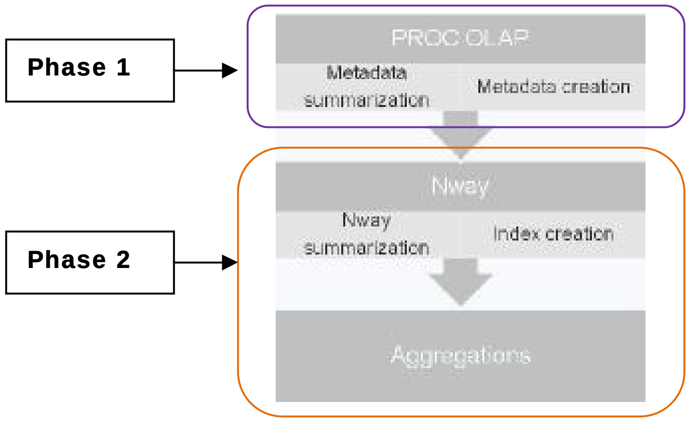


Figure 2 Phases of the OLAP cube build process

PROC OLAP – PHASE I

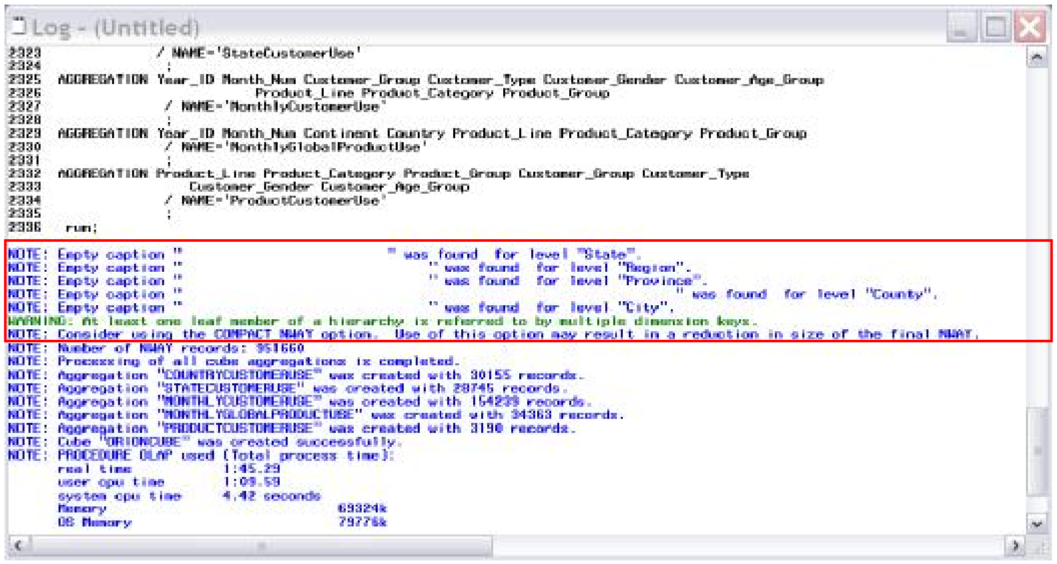
Phase I comprises the internal metadata preparation which includes metadata summarization and sorting of member infomation, and the creation of the internal cube metadata.

Cube metadata includes information about:

- location of the data
- the cube structure
- members and their relationship with each other for navigation purposes, formats, member properties
- calculated members

The first phase occurs immediately upon execution and completes just prior to the NWAY message relating to number of records. The log shown in Figure 3 is that of a cube build using a star schema input source, the first phase is highlighted with the red box.

Notes about “Empty caption,” are indicators that ragged and/or unbalanced hierarchies exist in this cube. These messages occur when members in levels are identified that match the EMPTY_NUM or EMPTY_CHAR option. In the example below, the EMPTY_CHAR=”” option was used on the PROC OLAP statement.



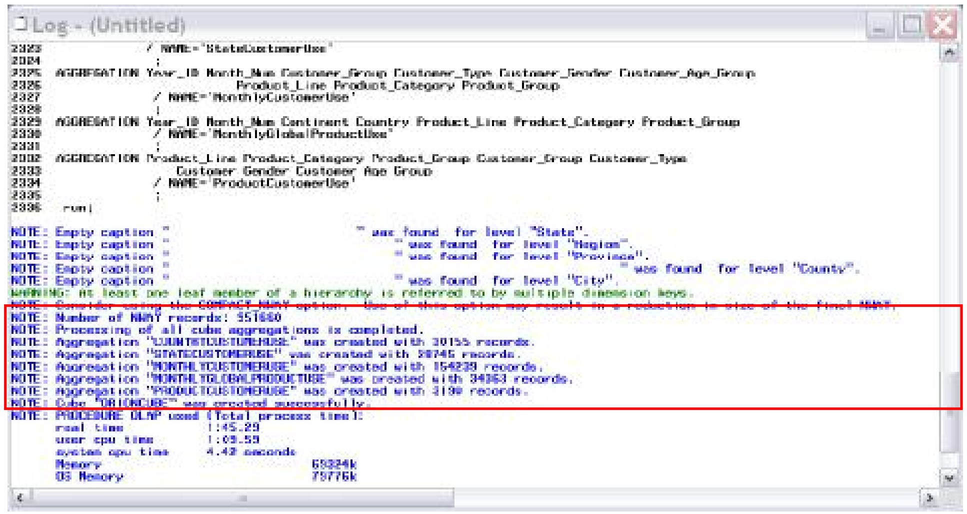
```
2323               / NAME='StateCustomerUse'
2324                ;
2325  AGGREGATION Year_ID Month_Num Customer_Group Customer_Type Customer_Gender Customer_Age_Group
2326                          Product_Line Product_Category Product_Group
2327                / NAME='MonthlyCustomerUse'
2328                ;
2329  AGGREGATION Year_ID Month_Num Continent Country Product_Line Product_Category Product_Group
2330                / NAME='MonthlyGlobalProductUse'
2331                ;
2332  AGGREGATION Product_Line Product_Category Product_Group Customer_Group Customer_Type
2333                   Customer_Gender Customer_Age_Group
2334                / NAME='ProductCustomerUse'
2335                ;
2336   run;

NOTE: Empty caption "                          " was found  for level "State".
NOTE: Empty caption "                              " was found  for level "Region".
NOTE: Empty caption "                              " was found  for level "Province".
NOTE: Empty caption "                                                    " was found  for level "County".
NOTE: Empty caption "                              " was found  for level "City".
WARNING: At least one leaf member of a hierarchy is referred to by multiple dimension keys.
NOTE: Consider using the COMPACT_NWAY option.  Use of this option may result in a reduction in size of the final NWAY.
NOTE: Number of NWAY records: 951660
NOTE: Processing of all cube aggregations is completed.
NOTE: Aggregation "COUNTRYCUSTOMERUSE" was created with 30155 records.
NOTE: Aggregation "STATECUSTOMERUSE" was created with 28745 records.
NOTE: Aggregation "MONTHLYCUSTOMERUSE" was created with 154239 records.
NOTE: Aggregation "MONTHLYGLOBALPRODUCTUSE" was created with 34363 records.
NOTE: Aggregation "PRODUCTCUSTOMERUSE" was created with 3190 records.
NOTE: Cube "ORIONCUBE" was created successfully.
NOTE: PROCEDURE OLAP used (Total process time):
      real time           1:45.29
      user cpu time       1:09.59
      system cpu time     4.42 seconds
      Memory                                693248k
      OS Memory                             797768k
```

Figure 3 PROC OLAP log with first phase of the cube build process identified

PROC OLAP – PHASE II

The second phase of build process is the creation of the NWAY and aggregation tables (and their related indices, if requested). The indices make data retrieval faster and more efficient, choosing to “not build” the indices will result in slower query performance and is not recommended. This second phase can be identified in log shown in Figure 4 with a red box and always starts just prior to the note “Number of NWAY records” and concludes with a message which indicates the successful cube build. (See Logging Section for further granularity in identifying the start of Phase II)



```
2323               / NAME='StateCustomerUse'
2324                ;
2325  AGGREGATION Year_ID Month_Num Customer_Group Customer_Type Customer_Gender Customer_Age_Group
2326                          Product_Line Product_Category Product_Group
2327                / NAME='MonthlyCustomerUse'
2328                ;
2329  AGGREGATION Year_ID Month_Num Continent Country Product_Line Product_Category Product_Group
2330                / NAME='MonthlyGlobalProductUse'
2331                ;
2332  AGGREGATION Product_Line Product_Category Product_Group Customer_Group Customer_Type
2333                   Customer Gender Customer Age Group
2334                / NAME='ProductCustomerUse'
2335                ;
2336   run;

NOTE: Empty caption "                          " was found  for level "State".
NOTE: Empty caption "                              " was found  for level "Region".
NOTE: Empty caption "                              " was found  for level "Province".
NOTE: Empty caption "                                                    " was found  for level "County".
NOTE: Empty caption "                              " was found  for level "City".
WARNING: At least one leaf member of a hierarchy is referred to by multiple dimension keys.
NOTE: Consider using the COMPACT_NWAY option.  Use of this option may result in a reduction in size of the final NWAY.
NOTE: Number of NWAY records: 951660
NOTE: Processing of all cube aggregations is completed.
NOTE: Aggregation "COUNTRYCUSTOMERUSE" was created with 30155 records.
NOTE: Aggregation "STATECUSTOMERUSE" was created with 28745 records.
NOTE: Aggregation "MONTHLYCUSTOMERUSE" was created with 154239 records.
NOTE: Aggregation "MONTHLYGLOBALPRODUCTUSE" was created with 34363 records.
NOTE: Aggregation "PRODUCTCUSTOMERUSE" was created with 3190 records.
NOTE: Cube "ORIONCUBE" was created successfully.
NOTE: PROCEDURE OLAP used (Total process time):
      real time           1:45.29
      user cpu time       1:09.59
      system cpu time     4.42 seconds
      Memory                                69324k
      OS Memory                             79776k
```

Figure 4 PROC OLAP log with the second phase of the cube build process identified

SAS OPTIONS

Below are specific SAS options that help optimize and control the cube build process at each phase. For more detail on each option please refer to the SAS 9.2 OLAP Server: Users Guide.

Option	Type	Description	Relevance
MEMSIZE	SYSTEM	Specifies the limit on the total amount of virtual memory that can be used by a SAS session.	PHASE I and PHASE II
REALMEMSIZE	SYSTEM	Specifies the amount of real (physical) memory SAS can expect to allocate.	PHASE I and PHASE II
SUMSIZE	SYSTEM	Specifies the limit on the memory allocation for the Summarization Process	PHASE I and PHASE II
SORTSIZE	SYSTEM	Specifies the limit on the memory allocation for the Sort Process	PHASE I and PHASE II
UTILLOC*n*	SYSTEM	Specifies one or more file system locations in which applications can store utility files. (WORK will be used if UTILLOC is not set)	PHASE I and PHASE II
SPDEUTILLOC	SYSTEM	Specifies one or more file system locations in which temporarily utility files can be written. (UTILLOC will be used if SPDEUTILLOC is not set)	PHASE II
INDEXSORTSIZE	PROC OLAP	INDEXSORTSIZE indicates the max amount of memory per thread of execution you want to make available during NWAY and aggregation index creation.	PHASE II
MAXTHREADS	PROC OLAP	Specifies the maximum number of threads that are used to asynchronously create aggregation indexes. Setting MAXTHREADS too high can lead to performance degradations.	PHASE II
ASYNCINDEXLIMIT	PROC OLAP	The ASYNCINDEXLIMIT option is new to SAS® 9.2. It gives the user the control over how many indexes for one table can be built at a time.	PHASE II
ADDNWAYINDEX	PROC OLAP	Ability to create NWAY indexes after the initial PROC OLAP cube build process, in which NOINDEX was used for the NWAY.	PHASE II
INDEX | NOINDEX	PROC OLAP	Specifies whether or not to create aggregations with indexes.	PHASE II
CONCURRENT	PROC OLAP	Specifies the maximum number of aggregations to create in parallel. It does not impact the NWAY which is always built first.	PHASE II
COMPRESS	PROC OLAP	Compress the resulting NWAY and aggregation tables, reducing the size of the cube.	PHASE II

Table 1 Relevant Options to control cubes build process

RECOMMENDATIONS

Input Data:

A star schema is the optimal data source for building a MOLAP cube. A star schema is comprised of dimension tables and a fact table which allows SAS to read and translate the information faster. Additional features are available only with a star schema such as multiple language support and more flexibility during cube update.

Avoid where possible using a data set view as an input data source as this requires additional system resources for processing the view and can cause contention with the resources that are required for the cube building process.

Memory Options:

MEMSIZE:

Memsize is an indicator of the total memory you want to allow for this SAS process. Setting this value to 0 (or MAX) is useful for debugging or testing purposes but highly discouraged in a production environment. You can use the fullstimer option in SAS combined with MEMSIZE=MAX to determine how much your cube build will actually need and then set your MEMSIZE slightly larger for your production builds. If you are building cubes concurrently you should adjust the MEMSIZE smaller to avoid contention for system resources.

REALMEMSIZE:

REALMEMSIZE should be a percentage of MEMSIZE but this will be dependent on the size of your data.

SUMSIZE:

Proper specification of SUMSIZE= can improve procedure performance by restricting the swapping of memory that is controlled by the operating environment. Generally, the value of the SUMSIZE= system option should be less than the physical memory available to your process. If the procedure you are using needs more memory than you specify, the system creates a temporary utility file. If the value of SUMSIZE is greater than the values of the MEMSIZE option and the REALMEMSIZE option, SAS uses the values of the MEMSIZE option and REALMEMSIZE option.

Temporary work space:

UTILLOC:

During the summarization phase, if both UTILLOC1 and UTILLOC2 are specified, they will each be used for two distinctly different functions of the summarization process. At a minimum both UTILLOC1 and UTILLOC2 should be big enough to hold the size of the input data (give a little more than just the exact size of the input table for each location) and UTILLOC locations should be of equal size.

For Phase II, SPDEUTILLOC (or UTILLOC) should be set to allow for a separate location for the temporary utility files needed in this phase.

It is strongly recommended that ALL temporary workspace locations be located on physically separate disks from that of the input data to avoid disk contention.

NWAY Creation:

ASYNCINDEXLIMIT:

This is most effective for the NWAY table which will contain indexes for all hierarchies in the cube. The default behavior at SAS 9.1.3 was unlimited. This meant that all indices were being created at one time which has lead at times to thread contention when indexing very large tables (such as the NWAY). It is recommended not to use this function when dealing with small data and low cardinality as it will not add to performance and may have an adverse interaction with the cube build process.

ADDNWAYINDEX:

It was added for cubes that completed successfully but the NWAY index build failed. At SAS® 9.2, if this occurred you would also want to consider bumping up your MEMSIZE and using the ASYNCINDEXLIMIT option described above.

General Aggregation Index Creation:

INDEXSORTSIZE/MAXTHREADS:

Typically MAXTHREADS should be set to a number less than the number of CPUs available. The default is calculated off of the CPUCOUNT option which is an “indicator” of the number of available CPUs (it is not a hard and fast limit). On UNIX machines at SAS® 9.2, CPUCOUNT defaults the min value of 4 and ACTUAL. Example: INDEXSORTSIZE=4096. If MAXTHREADS was set to 4, you would be telling the subsystem that each thread has 1G of memory it can use. The INDEXSORTSIZE applies to a certain subset of the memory usage during index creation. Additional memory is needed for other functionality at that time so it is important to keep that in mind when setting this option.

PROBLEMS AND RESOLUTIONS

Work Space (Utility) Problems:

a. If you get a message mentioning “Utility write failed” or “disk full condidtion” check both UTILLOC locations. If one location is full but the other is not, this is an indication that the size for both UTILLOC locations should be increased.

b. If you receive the error “Utility File read failed” make sure that nolargefiles is not set on the file systems as no file will be created greater than 2GB. This is specific to UNIX operating systems. You can validate if this option is set by checking /etc/mnttab.

c. If one of the UTILLOC locations has been set too small expect a system error (not a SAS error) in PHASE I during the data summarization step “ERROR: Complex utility space issue” or you may see the operating system CPU usage drop to ~1% and hang. Increase the UTILLOC location as per the recommendation mentioned above in PHASE I – Recommendations.

PHASE I (data summarization) Problems:

a. Errors with the terms “classification”, “summarization” or “category” are usually indicators that SUMSIZE needs to be increased. (and / or MEMSIZE and REALMEMSIZE).
Example:

```
ERROR: Out of memory.
ERROR: A problem was encountered during data summarization phase (8).
NOTE: The initial memory limit for classification tables was 1239026K bytes.
Actual memory acquired was 1244992K bytes.
```

b. If you are encountering a “Critical memory shortage” or “Memory exhausted message” during the data summarization step MEMSIZE and REALMEMSIZE need to be adjusted. To make sure that you can track memory usage during the summarization step use %let syssumtrace=3; Refer to the logging section below for more details.

PHASE II (NWAY and Aggregation creation) Problems:

a. If an error occurs after the message “IN PROGRESS: NWAY index creation” and it contains the keyword “index/es” and “NWAY aggregation” investigate using ASYNCINDEXLIMIT, INDEXSORTSIZE and tuning MEMSIZE to control a successful NWAY creation.

b. If you are adding aggregations to an existing cube for improving query performance and receive the below error messages or a message containing the words “No memory can be allocated” , “index” or “aggregation” and “aggregate”; resource contention may be the issue. Consider the interaction of PROC OLAP options MAXTHREADS=, CONCURRENT= and INDEXSORTSIZE= and reduce memory utilization.

```
ERROR: Insufficient memory
ERROR: A failure occurred as a result of a problem in creating cube
aggregations
ERROR: New aggregations cannot be created for the existing cube. All
AGGREGATION statements will be ignored
```

INCREMENTAL UPDATE

With the release of SAS 9.2 OLAP Server, you can now add new data to your cubes *without* rebuilding the entire cube via the incremental update process. An incremental update typically involves adding cell data and members to an existing cube as shown in Figure 5.

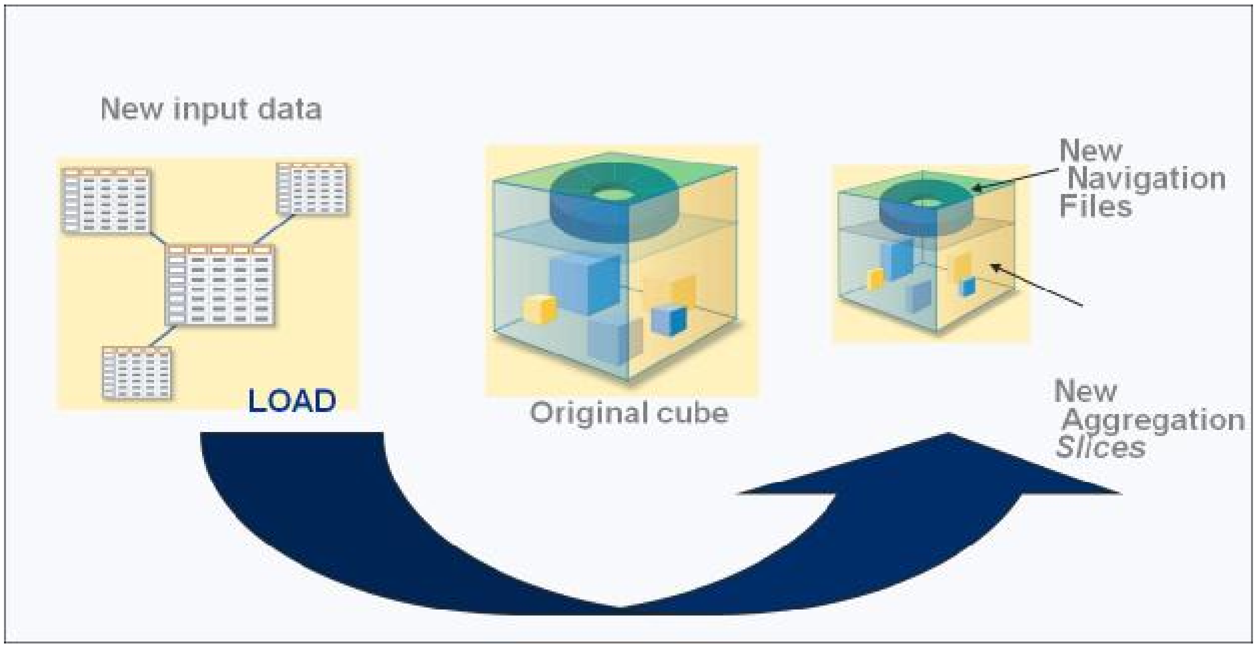


Figure 5 Incremental Update Process for an OLAP Cube and the addition of new data

There are several decisions at cube creation that can impact the cube build process for incremental updates. These were discussed at length in "Paper 330-2008 Avoid Growing Pains: New Cube Update Features You Should Know About" (Weinberger, Tierney). In the following sections, two key decisions will be discussed and the impact these decisions can make on Phase I and Phase II of the cube build process:

- Do all dimensions need updating (typically only some dimensions such as time will need updating)?
- What information is being updated (is it metadata information, actual data values or both)?

DO ALL DIMENSIONS NEED UPDATING?

We would recommend that you take time, prior to the initial build of your cube, to review your cube's dimensions. Identify any dimensions that contain fixed number of distinct values, these dimensions can be pre-loaded and the dimension can be marked NONUPDATEABLE. For example, a relatively static dimension would be Geography, while a dimension such as time or customer would require regular additions of new members.

Benefits

Smaller footprint:

The first is the smaller disk footprint. All aggregations that include this dimension will take up less space when this option is used.

NOTE: By default, dimensions are UPDATEABLE.

Table 2 illustrates the effect of the NONUPDATEABLE option specified on the PROC OLAP statement, on the overall disk footprint for that cube. With NONUPDATEABLE option used on PROC OLAP statement instead of the DIMENSION statement all dimensions will be stored in the non-updateable format.

Cube Name	All dimensions UPDATEABLE (MB)	All dimensions NON-UPDATEABLE (MB)	% Smaller
Cube1	209	173	17%
Cube2	142	89.4	37%
Cube3	386	244	37%
Cube4	360	220	39%
Cube5	261	116	56%

Table 2 Cube size comparisons with NONUPDATEABLE option

Update performance boost:

The second benefit to using the NONUPDATEABLE option is a performance boost during cube update. Dimensions that are ***updateable*** will have their dimension tables scanned for every update. ***Non-updateable*** dimension tables will never be re-scanned for potential new members.

WHAT INFORMATION IS BEING UPDATED?

As mentioned above, a cube comprises of metadata, navigation files and aggregations. Typically when updating a cube you will either be:

- updating the navigation files which contain member information (UPDATE_DIMENSION option)
- or the aggregations which contain the summarized data values (ADD_DATA option)
- or both.

ADD_DATA

The ADD_DATA option on the PROC OLAP statement enables you to add new members and data to a cube. You can update a cube in-place or create generations of the cube (see SAS 9.2 OLAP Server: Users Guide for more details). Here is an example of the ADD_DATA option used with PROC OLAP:

```
PROC OLAP data=mylib.newdata cube=cubeA add_data
outcube=cubeB outschema=testSchema;
metasvr host="myhost" port=8561 repository=myrepository
olap_schema=prodSchema;
run;
```

The ADD_DATA option requires that either UPDATE_IN_PLACE, OUTCUBE=, or OUTSCHEMA= also be specified.

UPDATE_DIMENSION

The UPDATE_DIMENSION option on the DIMENSION statement gives you the flexibility on the updateable dimensions to control whether or not the dimension table is scanned. By default if the dimension is updateable the table will be scanned during each update event (default: UPDATE_DIMENSION=MEMBERS)

The values for the UPDATE_DIMENSION statement are as follows:

MEMBERS:

This means that the dimension table currently associated with the dimension should be read and processed for new members of every hierarchy in that dimension. When specified on a star schema cube *without* the PROC OLAP **ADD_DATA** option, the MEMBERS option provides a way to update the cube with new members without having to add data to the cube.

MEMBERS_AND_PROPERTIES:

This means that the dimension table currently associated with the dimension should be read and processed for new members PLUS member properties for existing members should be changed with the new values in the dimension table. When specified on a star schema cube *without* the PROC OLAP **ADD_DATA** option, the MEMBERS_AND_PROPERTIES option provides a way to update the cube with new members and properties without having to add data to the cube.

OFF:

This means that the dimension table currently associated with the dimension should NOT be read at all. It applies only to cubes loaded from a star schema and never to a cube loaded from a single detail table. If the ADD_DATA option is used on the PROC, and an updateable dimension's table should not be processed, then this option must be used to avoid a table scan. If the ADD_DATA option is not used on the PROC, then **OFF** is the DEFAULT option for all dimensions in the cube.

INCREMENTAL UPDATE - PHASE I

Remember that the two steps of PHASE I cube build process is metadata summarization and metadata creation. In incremental update the process is similar – metadata summarization of the new member information occurs and then the existing member metadata is updated.

There are instances, however, where neither of these steps is performed when updating a cube. If the cube was originally created using the NONUPDATEABLE option on the PROC OLAP statement then PHASE I will not be needed.

Also, PHASE I is not performed for specific dimensions in an incremental update when the DIMENSION statement has either NONUPDATEABLE or UPDATE_DIMENSION=OFF option specified.

INCREMENTAL UPDATE – PHASE II

If the option ADD_DATA is specified then the new data is read. An NWAY and aggregation tables are created for that new data and is added as a slice of the cube as depicted in Figure 5.

LOGGING OPTIONS

There are many different logging options available through SAS, for general logging information refer to the "Paper 304-2009 The SAS OLAP Server: Understanding and Solving Common Problems" (Budlong).

The following table describes additional logging options that our customers have found particularly useful in gaining a better understanding of the build process and diagnosing issues when they occur.

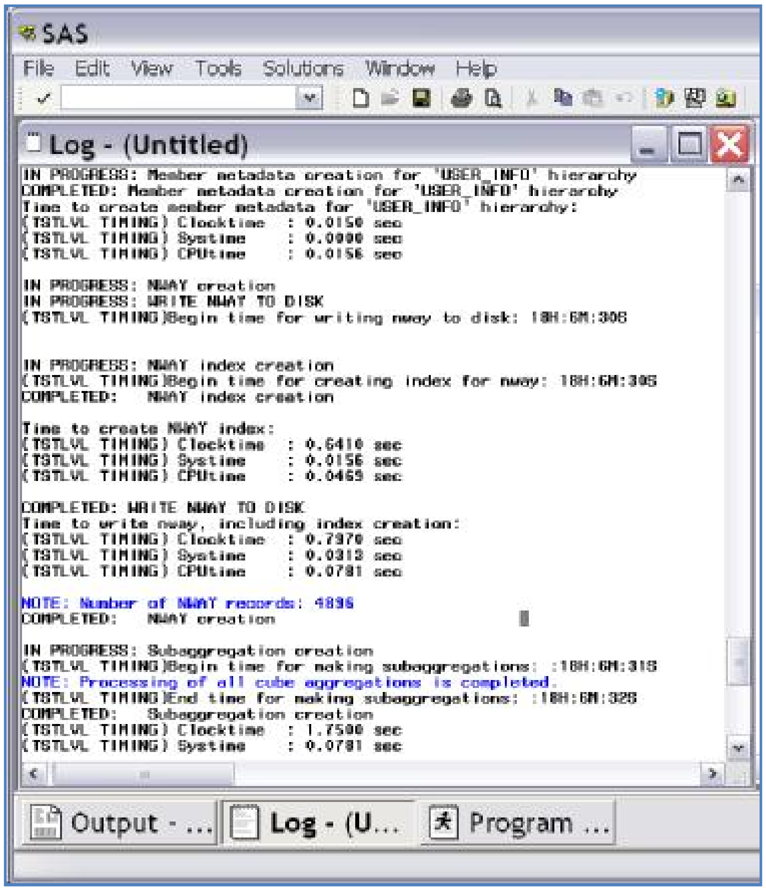
Logging Option and Description	Usage	Sample output

SYSSUMTRACE
 	
Turns on additional logging for PHASE I of the PROC OLAP process. Showing summarization memory usage.
	`%let syssumtrace=3;`	`IN PROGRESS: WRITE NWAY TO DISK`
		`(TSTLVL TIMING)Begin time for writing nway to disk: 14H:45M:1S`
		`NOTE: The initial memory limit for classification tables was 1219133K bytes. Actual memory acquired was 2368K bytes.`
		`NOTE: The utility file buffer size selected was 16K bytes.`

PROC OPTIONS (MEMORY)
 	
Get information about MEMORY settings and how they got their values – from the command line, shipped default, config file etc.
	`PROC OPTIONS`
	`  group=`**`MEMORY value`**`;`
	`run;`
		`Option Value Information For SAS Option MEMSIZE`
		`    Option Value: 2147483648`
		`    Option Scope: SAS Session`
		`    How option value set:  SAS Session Startup Command Line`

PROC OPTIONS (PERFORMANCE)
 	
Get information about PERFORMANCE settings and how they got their values.
	`PROC OPTIONS`
	` group=`**`PERFORMANCE value`**`;`
	`run;`
		`Option Value Information For SAS Option SPDEUTILLOC`
		`    Option Value:`
		`    Option Scope: Default`
		`    How option value set:  Shipped Default`

TEST_LEVEL=2
 	
Gives finer grained logging of the cube build process.
 	
The *actual* beginning of Phase II is clearly demarcated, by the test _level message:
 	
`IN PROGRESS: NWAY creation`
 	
This message occurs well before the first NOTE in the log about the NWAY.
 	
PLEASE NOTE! TEST_LEVEL is an undocumented option of the PROC OLAP statement
	`PROC OLAP CUBE=name`
	`TEST_LEVEL=2`
	
	`<additional PROC OLAP statement options>;`
		General TEST_LEVEL log format
		PHASE I
		o Connect with metadata server
		o Initialize cube build
		o Per hierarchy –
		• get captions
		• create member metadata
		• elapsed times
		PHASE II
		o N WAYcreation begins
		o NWAY write to disk
		o NWAY index build
		o Aggregation builds begin (aka subaggregation)
		o Cube files disked
		o Metadata server updated with cube information
		o Elapsed times
		
		See Figure **6** for “Sample Output”



```
IN PROGRESS: Member metadata creation for 'USER_INFO' hierarchy
COMPLETED: Member metadata creation for 'USER_INFO' hierarchy
Time to create member metadata for 'USER_INFO' hierarchy:
(TSTLVL TIMING) Clocktime   : 0.0150 sec
(TSTLVL TIMING) Systime     : 0.0000 sec
(TSTLVL TIMING) CPUtime     : 0.0156 sec

IN PROGRESS: NWAY creation
IN PROGRESS: WRITE NWAY TO DISK
(TSTLVL TIMING)Begin time for writing nway to disk: 18H:6M:30S


IN PROGRESS: NWAY index creation
(TSTLVL TIMING)Begin time for creating index for nway: 18H:6M:30S
COMPLETED:   NWAY index creation

Time to create NWAY index:
(TSTLVL TIMING) Clocktime   : 0.6410 sec
(TSTLVL TIMING) Systime     : 0.0156 sec
(TSTLVL TIMING) CPUtime     : 0.0469 sec

COMPLETED: WRITE NWAY TO DISK
Time to write nway, including index creation:
(TSTLVL TIMING) Clocktime   : 0.7970 sec
(TSTLVL TIMING) Systime     : 0.0313 sec
(TSTLVL TIMING) CPUtime     : 0.0781 sec

NOTE: Number of NWAY records: 4896
COMPLETED:   NWAY creation

IN PROGRESS: Subaggregation creation
(TSTLVL TIMING)Begin time for making subaggregations: :18H:6M:31S
NOTE: Processing of all cube aggregations is completed.
(TSTLVL TIMING)End time for making subaggregations: :18H:6M:32S
COMPLETED:   Subaggregation creation
(TSTLVL TIMING) Clocktime   : 1.7500 sec
(TSTLVL TIMING) Systime     : 0.0781 sec
```

Figure 6 Sample output of TEST_LEVEL=2 option

CONCLUSION

By understanding the components of a MOLAP cube, metadata and data, the cube build process can be explained using two Phase approach. PHASE I is associated to processing the metadata, and PHASE II the processing of the data. Being aware of the appropriate SAS options and PROC OLAP options and how they can be applied can:

- Ensure a successful build especially when assigning memory consumption and disk space
- Optimization the build processes, maximizing the delivery of the data while minimizing the impact to your system

In addition, recommendations have been made so that you can design, architect and recognize problems with your build process, allowing you to make decisions relating to:

- Input data
- Temporary work space
- NWAY creation
- Aggregation creation

Ultimately, this paper will aid in the success of building your cube therefore, providing your business user information in a timely manner. Allowing the business user to focus on the right business decision they need to make for your company.

ACKNOWLEDGMENTS

Thank you to all who helped out with testing and contributing to the content of this paper.

RECOMMENDED READING

SAS® 9.2 OLAP Server: User's Guide

Paper 330-2008 Avoid Growing Pains: New Cube Update Features You Should Know About (Weinberger, Tierney)

Paper 304-2009 The SAS® OLAP Server: Understanding and Solving Common Problems (Budlong)

CONTACT INFORMATION

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

Mary Simmons
SAS Institute Inc.
Campus Drive
Cary, NC, 27513
Email: mary.simmons@sas.com
Web: www.sas.com

Michelle Wilkie
SAS Institute Inc.
Campus Drive
Cary, NC, 27513
E-mail: michelle.wilkie@sas.com
Web: www.sas.com