SAS® 9.1.3 OLAP Server

User's Guide
Second Edition

THE POWER TO KNOW®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *SAS® 9.1.3 OLAP Server: User's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

SAS® 9.1.3 OLAP Server: User's Guide, Second Edition

ISBN 13: 978-1-59047-821-9
ISBN 10: 1-59047-821-5

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, March 2006
2nd printing, October 2006

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents

What's New

Overview

The SAS OLAP Server enables users to develop and deploy scalable Online Analytical Processing (OLAP) applications. In addition, automated data loading and cube building are available through the use of a new administration interface called SAS OLAP Cube Studio, which was developed using Java technology.

OLAP queries are performed using the Multidimensional Expressions (MDX) query language in client applications that are connected to the SAS OLAP Server by using the following:

- the SQL Pass-Through Facility for OLAP, which is designed to process MDX queries within the PROC SQL environment
- open access technologies such as OLE DB for OLAP, ADO MD, and Java

New and enhanced features in SAS OLAP Server include the following:

- new tools for data loading and cube building
- new options added to the PROC OLAP statement
- new functions
- new tuning capabilities for the query thread pool
- new tuning options window
- improved performance
- improved querying capability
- improved aggregation tuning
- additional enhancements

Note: This section describes the features of the SAS OLAP Server that are new or enhanced since SAS 8.2. △

Documentation Enhancements

The *SAS OLAP Server: Administrator's Guide* is no longer available under that title. The content of that document has been merged into the new administrative document

set for SAS Intelligence Platform. The other SAS OLAP Server documents, the *SAS OLAP Server: User's Guide* and the *SAS OLAP Server: MDX Guide* remain available under their existing titles.

The content of the *SAS OLAP Server: Administrator's Guide* is now available in the following documents:

SAS Intelligence Platform: Application Server Administration Guide
describes the configuration, tuning, and management of SAS OLAP Servers, along with the other servers that are defined as part of the logical SAS Application Server.

SAS Intelligence Platform: Desktop Application Administration Guide
describes the administration of SAS OLAP Cube Studio.

SAS Intelligence Platform: System Administration Guide
describes SAS OLAP Server monitoring, start/stop/restart, cube data import/export.

SAS Intelligence Platform: Security Administration Guide
describes security issues for SAS OLAP cubes, including a newly updated section about member-level security.

SAS Intelligence Platform: Data Administration Guide
describes how to define libraries and schemas for SAS OLAP cubes.

The administrative document set for the SAS Intelligence Platform is available in the SAS Online Documentation at *http://support.sas.com/onlinedoc/913/docMainpage.jsp*. The home page for the administrative document set is available at *http://support.sas.com/documentation/configuration/913admin.html*.

New Tools for Data Loading and Cube Building

The OLAP procedure, in addition to cube building, includes options for handling ragged hierarchies, defining global calculated members and named sets, assigning properties to levels, and optimizing cube creation and query performance. It also supports multiple hierarchies and drill-through tables.

SAS OLAP Cube Studio is an alternative Java interface to the OLAP procedure. This interface is also integrated with SAS Data Integration Studio.

New Options Added to the PROC OLAP Statement

The following options have been added to the PROC OLAP statement:

COMPACT_NWAY
when building a cube from a star schema, this option enables additional summarizations during the cube build that can decrease the size of the NWAY aggregation.

IGNORE_MISSING_DIMKEYS=TERSE | VERBOSE
when building a cube from a star schema, this option enables the continuation of the build when the fact table is found to contain keys that are not present in any of the dimension tables. The log receives information about the number and location of the missing keys.

For more information about these new options, see the *SAS OLAP Server: User's Guide*.

New Functions

The following functions are new:

- A Specify Map function has been added to SAS OLAP Cube Studio. The Specify Map function enables you to store ESRI Geographic Information System (GIS) spatial map information in the SAS Metadata Repository. This GIS information can then be read by the SAS OLAP Server and returned during a cube query.
- The Export Cube and Import Cube functions enable you to copy cube metadata from a source repository to a target repository, and if needed, to another server. The Export Cube function enables you to extract a cube's metadata from the source repository and save it in a file that you specify. All information about a cube, including its dimensions, hierarchies, levels, measures, notes, properties, calculated measures, aggregations, and security settings, will be extracted. The Import Cube function then enables you to save the cube metadata to another metadata repository on another metadata server.
- The Define Distinct Count function enables you to store NUNIQUE (Distinct Count) statistics as measures with a SAS OLAP cube. You can add or delete a distinct count measure either with PROC OLAP or in the Cube Designer wizard, within SAS OLAP Cube Studio. You can add or delete a Distinct Count measure. You can also modify the name, caption, format, units, and description of the measure.
- The Synchronize Levels function enables you to synchronize a cube when the input table for an existing cube has encountered a column name change. This function finds the name differences between the cube and its input table and changes the hierarchy level names to match the input table column names.

New Tuning Capabilities for the Query Thread Pool

New tuning capabilities for the query thread pool are now available for each of your SAS OLAP Servers. Tuning the query thread pool enables you to optimize performance based on the number and frequency of query requests that are received by a SAS OLAP Server.

New Tuning Options Window

A tuning Options window has been added to the Manual Tuning function and the Advanced Aggregation Tuning plug-in. The Options window enables you to specify certain performance options on the PROC OLAP statement when adding or modifying aggregations for a cube. In this window you can view the current values and change the values when needed.

Improved Performance

Server performance is recorded and analyzed by using the Application Response Measurement (ARM) system.

The new multi-threaded data storage and server functionality provide faster cube performance. The data can be stored in a multidimensional form (MOLAP) or in a form that includes existing aggregations from presummarized data sources.

Improved Querying Capability

An SQL Pass-Through Facility for OLAP is available in SAS for use in querying cubes.

A subquery cache can now be enabled, disabled, and sized for each of your SAS OLAP Servers. The new cache stores subqueries that generate empty result sets.

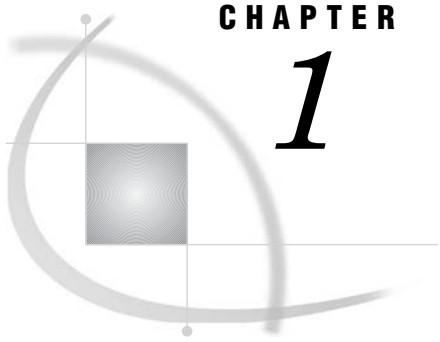
Improved Aggregation Tuning

Aggregations can be added to or deleted from existing cubes. In addition, further enhancements for aggregation tuning have been made:

- The AGGREGATION statement and SAS OLAP Cube Studio's Manual Tuning function have been enhanced for use with cubes that employ aggregated data from other tables or for use with cubes that have no NWAY aggregation.
- The Advanced Aggregation Tuning plug-in provides a point-and-click interface that enables you to create and add aggregations to the list of existing aggregations that might already be defined for the cube.

Additional Enhancements

- The metadata structure is improved, and metadata is stored with the cube.
- Caching and logging can be enabled or disabled.
- Support for ad hoc calculations and time dimensions is improved.
- The start-up file generator has been removed from the SAS OLAP Server Monitor plug-in for SAS Management Console. SAS OLAP Servers are now automatically installed as Windows services; server start-up options are set by an INI file that is installed by the SAS Deployment wizard.
- When creating a cube dimension in SAS OLAP Cube Studio's Cube Designer wizard, a hierarchy for a dimension will automatically be created when no hierarchy has been explicitly defined and you click the **`Finish`** button in this window. This default hierarchy includes all levels that were specified for the current dimension and the order they were listed in for the dimension.
- The Calculated Members plug-in provides a point-and-click interface that enables you to add new measures or modify existing measures for the selected cube.



CHAPTER

1

OLAP Introduction and Overview

What Is OLAP?

Online Analytical Processing (OLAP) is a technology that is used to create decision support software. OLAP enables application users to quickly analyze information that has been summarized into multidimensional views and hierarchies. By summarizing predicted queries into multidimensional views prior to run time, OLAP tools provide the benefit of increased performance over traditional database access tools. Most of the resource-intensive calculation that is required to summarize the data is done before a query is submitted.

Data Storage and Access

Decision makers are asked to make timely and accurate decisions that are based on the past performance and behavior of an organization as well as on future trends and directives. To make effective business decisions, business analysts must have access to the data that their company generates and responds to. This access must include timely queries, summaries, and reviews of numerous levels and combinations of large, recurrent amounts of data. The information that business analysts review determines the quality of their decisions.

Organizations usually have databases and data stores that maintain repeated and frequent business transaction data. This provides simple yet detailed storage and retrieval of specific data events. However, these data storage systems are not well suited for analytical summaries and queries that are typically generated by decision makers. For decision makers to reveal hidden trends, inconsistencies, and risks in a business, they must be able to maintain a certain degree of momentum when querying the data. An answer to one question usually leads to additional questions and review of the data. Simple data stores do not successfully support this type of querying.

A second type of storage, the data warehouse, is better suited for this. Data is maintained and organized so that complicated queries and summaries can be run. OLAP further organizes and summarizes specific categories and subsets of data from the data warehouse. This results in a robust and detailed level of data storage with efficient and fast query returns. SAS OLAP cubes can be built from either partially or completely denormalized data warehouse tables. Stored, precalculated summarizations called *aggregations*, can be added to the cube to improve cube access performance. Aggregations can either be pre-built relational tables, or you can let the cube create its own optimized aggregates.

Benefits of OLAP

The ability to have coherent and relevant information is the reason OLAP has gained in popularity. OLAP systems help reveal evasive inconsistencies and trends in data that might not have been seen before. OLAP users can intuitively search data that has been consolidated and summarized within the OLAP structure. In addition, OLAP tools allow for tasks such as sales forecasting, asset analysis, resource planning, budgeting, and risk assessment. OLAP systems also provide the following benefits:

- fast access, calculations, and summaries of an organization's data
- support for multiple user access and multiple queries
- the ability to handle multiple hierarchies and levels of data
- the ability to presummarize and consolidate data for faster query and reporting functions
- the ability to expand the number of dimensions and levels of data as a business grows.

To fully understand the benefits of OLAP and the details of its effective implementation, it helps to examine the technology from two perspectives—first, from that of the users and second, from that of the information technology (IT) administrators who are responsible for OLAP implementation. The users, typically business analysts and executives, expect the data to be organized according to categories that reflect the way in which they think about the enterprise. For IT administrators, OLAP can present a long list of technical issues, including these concerns:

- storage requirements and associated costs
- client and server capabilities

- maintenance activities such as update and backup
- performance considerations such as the amount of time that is required to build a multidimensional model
- the ability of the OLAP solution to integrate with current or planned data warehouse strategies and architectures.

OLAP Variations

OLAP technology can be further defined by the methods for storing and accessing data and by the performance of queries against that data. SAS OLAP supports three different variations of OLAP technology:

- MOLAP
- ROLAP
- HOLAP

MOLAP — Multidimensional OLAP

MOLAP (multidimensional online analytical processing) is a type of OLAP that stores summaries of detail data (aggregates) in multidimensional database structures. MOLAP cubes are most suited for slicing and dicing of data and are used when performance and query speed is critical. A unique feature of MOLAP is that calculations are predetermined and created with the cube. Whereas MOLAP is well suited for complex queries and calculations, it is limited in the amount of data it can handle.

ROLAP — Relational OLAP

ROLAP (relational online analytical processing) is a type of OLAP in which multidimensional data is stored in a relational database such as a SAS table or an ORACLE table. ROLAP is more scalable than other OLAP types and handles extensive amounts of data well. Although performance can be somewhat slow, ROLAP is limited only by the size of the relational database it is identified with.

ROLAP data is stored in either a flat file or with a star schema. With ROLAP, each instance of slicing and dicing of data is part of an SQL query (or multiple SQL queries) and is comparable to a WHERE clause in the SQL statement.

HOLAP — Hybrid OLAP

HOLAP (hybrid online analytical processing) is a type of OLAP in which relational OLAP (ROLAP) and multidimensional OLAP (MOLAP) are combined. In HOLAP, the source data is usually stored using a ROLAP strategy, and aggregations are stored using a MOLAP strategy. It combines the best features of both ROLAP and MOLAP. This combination usually results in the smallest amount of storage space. In HOLAP, aggregates can be precalculated and can be linked into a hybrid storage model.

What Is a Cube?

One of the advantages of OLAP is how data and its relationships are stored and accessed. OLAP systems house data in structures that are readily available for detailed queries and analytics. Cubes are central to the OLAP storage process.

A *cube* is a set of data that is organized and structured in a hierarchical, multidimensional arrangement. The cube is usually derived from a subset of a data

warehouse. Unlike relational databases that use two-dimensional data structures (often in the form of columns and rows in a spreadsheet), OLAP cubes are logical, multidimensional models that can have numerous dimensions and levels of data. Also, an organization typically has different cubes for different types of data.

One of the challenges of OLAP cube data storage and retrieval is the growth of data and how that growth affects the number of dimensions and levels in a cube hierarchy. As the number of dimensions increases over time, so does the number of data cells on an exponential scale. To maintain the efficiency and speed of the OLAP queries, the cube data is often presummarized into various consolidations and subtotals (aggregations).

Note: The SAS OLAP Server term *cube* is synonymous with the terms *hyper-cube* and *multi-cube*. △

Understanding the Cube Structure

OLAP cubes organize data in a hierarchical arrangement. Data is structured according to dimensions and measures.

Dimensions group the data along natural categories. (Examples of dimensions are Time, Products, Organization). Typically, dimensions offer different levels of grouping (for example, the Time dimension can be grouped by Years, Months, Days, etc.). *Levels* are organized into one or more hierarchies, typically from a coarse-grained level (for example, Year) down to the most detailed one (for example, Day). The individual category values (for example, 2002 or 21Jan2002) are called *members*.

Measures are the data values that are summarized and analyzed. Examples of measures are sales figures or operational costs. The data for measures is located in *cells*. Cells are the intersection of one member for every dimension.

Presummarized data in a cube is stored in aggregations. Aggregations are the basis for fast response to data queries in OLAP applications. An aggregation is possible at each intersection of a level of one or more dimensions. The selection of aggregations to presummarize is one of the major factors that determine query response time and cube size.

SAS Servers

SAS Servers and SAS OLAP

SAS OLAP uses a combination of SAS servers to store cube metadata, to store the physical cube structure, and to query cubes after they are created. Several types of SAS servers are available to handle different workload types and processing intensities. The term *server* refers to a program or programs that wait for and fulfill requests from client programs for data or services.

The SAS servers use the SAS Integrated Object Model (IOM), which is a set of distributed object interfaces that make SAS software features available to client applications when SAS is executed on a server. Each server uses a different set of IOM interfaces and has a different purpose. The following servers are used by SAS OLAP:

- the SAS Metadata Server
- the SAS Workspace Server
- the SAS OLAP Server

In addition to the other servers that SAS OLAP uses to store and process cube data, the SAS Stored Process Server is used in several different client applications that access SAS OLAP cubes and report on OLAP cube data.

SAS Metadata Server

The SAS Metadata Server controls access to a central repository of metadata that is shared by all of the applications in the system. It enables all users to access consistent and accurate data. The SAS Metadata Server stores the metadata that defines the cubes. It is a multi-user server that enables users to manage metadata in one or more metadata repositories by using the SAS Open Metadata Interface.

Note: The SAS Open Metadata Interface is an object-oriented application programming interface (API) that interacts with the SAS Metadata Server. SAS OLAP Cube Studio is an example of an application that is compliant with the SAS Open Metadata Interface. △

The SAS Metadata Server uses the Integrated Object Model (IOM) that is provided by SAS Integration Technologies. IOM provides distributed object interfaces to Base SAS software features. It enables you to use industry-standard languages, programming tools, and communication protocols to develop client programs that access these services on IOM servers.

In the SAS OLAP Server, all relevant structural information is contained within the cube and most of it is also replicated within the SAS Open Metadata Architecture. This is done so you can do the following:

- disassociate the cube definition process from cube creation, thus enabling you to create a cube by using its stored definition
- define and enforce security at the SAS Open Metadata Architecture level
- manage and control the data source in the centralized SAS Metadata Repository

You can find documentation about the SAS Open Metadata Architecture in the *SAS Intelligence Platform: System Administration Guide* and the *SAS Intelligence Platform: Data Administration Guide*.

SAS Workspace Server

The SAS Workspace Server enables client applications to submit SAS code to a SAS session using an application programming interface (API). The SAS Workspace Server provides access to SAS software features such as the SAS language, SAS libraries, the server file system, results content, and formatting services.

A program called the object spawner runs on a workspace server's host machine. The spawner listens for incoming client requests and launches server instances as needed. You can run as many instances of workspace servers as are needed to support your workload.

SAS OLAP Server

SAS OLAP Server is a scalable server that provides multi-user access to the data that is stored in SAS OLAP cubes. This server is designed to reduce the load on traditional back-end storage systems by quickly delivering summarized views, irrespective of the amount of data that underlies the summaries.

Processing data by using a multi-threaded kernel enables you to take advantage of your server's parallel processing abilities. SAS OLAP Server accepts data queries in the

industry-standard MDX query language, which opens it up to a variety of clients. The following features are also included:

- the SAS OLAP Cube Studio user interface, which is an alternative Java interface for building and maintaining cubes
- PROC OLAP for programmatically building and maintaining cubes
- server management by using SAS Management Console
- support for processing external aggregates
- support for OLE DB for OLAP

Note: OLAP queries are performed by using the Multidimensional Expressions (MDX) query language in client applications that are connected to the OLAP server by using OLE DB for OLAP (an extension of OLE DB that is used by COM-based clients), or through a similarly designed Java interface. △

SAS Stored Process Server

In addition to the other servers that SAS OLAP uses to store and process cube data, the SAS Stored Process Server is used to execute SAS Stored Processes. A stored process is a SAS program that is stored on a server and can be executed as required by requesting applications. SAS Stored Processes can be used in several different client applications that access SAS OLAP cubes and report on OLAP cube data, including the following:

- SAS Enterprise Guide
- SAS Data Integration Studio
- SAS Information Map Studio
- SAS Web Report Studio

The ability to store your SAS programs on the server provides an effective method for change control management. For example, instead of embedding the SAS code into client applications, you can centrally maintain and manage this code from the server. This gives you the ability to change your SAS programs and at the same time ensure that every client that invokes a stored process will always get the latest version available.

Stored process servers have MultiBridge connections, which enable multiple processes on different ports of the same server. A program called the object spawner runs on a stored process server's host machine. The spawner listens for incoming client requests and launches server instances as needed.

Why You Should Use Cubes

SAS cubes are designed to offer efficient data storage, fast data access, easy data maintenance, and flexibility in data management. The following sections explore cubes and multidimensional storage.

Cube Usage and Storage Space Reduction

While cubes are the format of choice to guarantee fast query response times against your data warehouse, SAS OLAP cubes are also often a very space efficient choice for data storage. In many cases, a basic cube without additional aggregations can be smaller than the input data because the process of creating the cube consolidates

records. SAS OLAP cubes use the hierarchy information for efficient aggregations storage. SAS OLAP cubes also deal efficiently with data sparsity by using virtual placeholders for empty cells. This removes the need for any physical representation of empty cells. A good rule of thumb is, the larger your input data, the greater the storage gain by loading data into a cube.

Multi-Threading Capabilities

Loading data into cubes and executing queries against the cube take advantage of the multi-threading capabilities of your server machine. Aggregations are created in parallel at cube build time. The creation of individual aggregations takes advantage of the Parallel Group-By capabilities of SAS' data engine. At query execution, the multi-threading capabilities of your server machine are fully used to concurrently serve queries by multiple users. Both query evaluation and data access are executed in parallel. To further increase query performance and reduce disk access, you can allocate additional memory on your server to be used for an in-memory aggregation cache.

Easy Setup and Maintenance

A cube is the physical representation of your logical dimensional model. The tools that are provided to update and maintain the cube reflect the multidimensional model, which makes both setup and maintenance of your cube as intuitive as possible. SAS' thin-client, Web-based administrator interface, SAS Management Console, enables you to set up and manage OLAP servers. SAS OLAP Cube Studio provides the workspace and cube designer tools that you need to create and maintain cubes. You can also use the SAS OLAP procedure to create and maintain cubes in a batch environment.

Data Management: Choosing Your Own Tool

If you create your own aggregations by using data management tools such as SQL, PROC SUMMARY, or the tools of your preferred relational database management system (RDBMS), then you can link those aggregations to your cubes without replicating the data within the cube. Any queries against those aggregations are executed by the appropriate SQL engine, and take advantage of any capabilities that engine might have. This allows you the flexibility to use the data management tools of your choice. It also allows you to distribute your data for your cube aggregations across multiple database systems, servers, and platforms. If you choose to let the cube builder create the aggregations, then you can control where to store the data and index files for each aggregation.

Analyzing Your Data

Data Preparation and Dimension Design

The goal of an OLAP system is to have data that is organized, available, and presented as relevant information to decision makers. OLAP cubes are based on data from data warehouses. A data warehouse consists of data that is extracted from

transactional systems at regular intervals. The extraction process works very closely with data quality control, making sure that the data is complete and accurate. Extensive data cleansing (which includes eliminating variant spellings of names) can be part of this task.

Building a data warehouse also implies transforming data that is optimized for transactional processing into data that is optimized for user-driven analysis. Part of that process is grouping facts and attributes into entities that correspond to the users' view of the organization. These groupings are known as dimensions. For related information, see the SAS online documentation for SAS Data Integration Studio.

An established technique for implementing a dimensional model is to create star join schemas that are based on the data. SAS OLAP cubes can be loaded from star schemas, or from further denormalized tables or views that include some or all dimensions in the fact table.

Data Tables Used to Define SAS OLAP Cubes

Detail Tables

A detail, or base, table is any table defined in a SAS Metadata Repository that contains the measures and levels for a cube. The detail table consists of unsummarized data that must include one column for each level and one numeric analysis column for each set of measures that will be generated.

Fact Tables and Dimension Tables

A star schema uses a set of input tables that are defined in the SAS Metadata Repository. The set of tables includes a single fact table and one or more dimension tables. The fact table must contain one numeric analysis column for each set of measures that will be generated. For levels, the fact table will either contain the columns for the levels of a dimension or it will contain a key column that links the fact table with a dimension table that contains the columns for the levels of a dimension. These statements are also true for star schemas:

- A dimension can be in the fact table. In this case, all the level columns are in the fact table and no fact or dimension key is required.
- If the dimension levels are defined in a dimension table, all the level columns for that dimension must be contained in the same dimension table.
- Both the dimension keys and fact keys are single columns, not combinations of columns.
- The dimension key can also be a level in the dimension.

Aggregation Tables

Aggregation tables are fully summarized external relational tables that are used to build cubes. All aggregation tables must contain a column for each measure in the cube where the statistic for the measure is one of the following: N, NMISS, SUM, MAX, MIN, or USS. Columns for derived measures cannot be stored on the aggregation table and are ignored if they exist. Derived measures are always computed at query time. An aggregation table can be used in two ways:

- as an NWAY data source for the cube. In this case, the table must contain a column for every level in the cube and a column for every stored measure.

- as a subaggregation for the cube. In this case, the table must include a column for each level of the aggregation and a column for every stored measure.

Drill-Through Tables

Drill-through tables are views maintained by the user that represent all of the input data used to define a cube. The drill-through table name can be set either when the cube is first created or during an update. Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source.

Aggregation Design

Efficient drilling or traversing of the cube data is a key factor in flexible and quick decision making and analysis. In order to maintain speed and consistency in reporting, data is usually precalculated or aggregated. An important factor in query performance is good aggregation design, which includes decisions about total storage space, available build time, storage location, and storage format.

When planning your data storage and design, it is helpful to approximate the size of aggregations. A basis for estimating aggregation size is the number of distinct values in a dimension level, otherwise known as cardinality. The other factor that determines aggregations size is density. Density is a measure of how many members of each dimension in an aggregation occur in combination with the members of the other dimensions (for example, there might not be sales of a specific product on a specific date). The total cube size as well as the resources that are available for the cube build process determine the build time that is needed. It is also important to note that build time should not exceed the cube update interval.

Aggregation size and available hardware influence your choices for aggregation partitioning. You can separate aggregations into multiple files. A reduced file size might accelerate OLAP server access time, particularly if multiple processors are available for multi-threaded processing. You can use preaggregated summary tables, the cube's own efficient aggregation storage, or a combination of both. Using indexes on either storage type might increase query performance, while also increasing storage space and build time.

After an initial aggregation design is chosen, subsequent cube builds enable you to optimize the cube's performance and size by adding or removing aggregations. You can analyze the OLAP users' behavior by using Application Response Measurement (ARM) logs, showing which aggregations are needed most and would be the most efficient.

CHAPTER

2

Building Cubes

Background

At this point in the cube-building process, the collecting and scrubbing of the data should be finished as well as planning a dimensional design. After you have collected and analyzed your data, you are ready to create the cube. When you define the cube, you define the dimensions and measures for the cube along with information about how aggregations should be created and stored. There are two methods of creating a cube:

- You can submit PROC OLAP code by using either the SAS Program Editor or a batch job. If you use PROC OLAP, the cube is created, and then the cube definition is stored in a metadata repository. This is referred to as the "long" form of PROC OLAP.
- You can use the Cube Designer interface in SAS OLAP Cube Studio to define and create the cube. The Cube Designer first stores the cube definition in a metadata repository, and then submits a shorter form of PROC OLAP code to create the cube. This is referred to as the "short" form of PROC OLAP.

Note: The Cube Designer can also be launched from *SAS Data Integration Studio*. △

Preparations for Building a Cube

To build a cube by using either PROC OLAP or SAS OLAP Cube Studio, you must complete several preliminary tasks:

- Configure a metadata server.
- Define an OLAP server in the metadata. The server does not need to be running to create cubes, but it must be defined in the metadata.
- Analyze the data to determine the location of the table(s) that will be used to build your cubes and what dimensions and measures will be created.
- Define the table(s) that will be used to create the cube in the metadata. You do this by using SAS Data Integration Studio or by using SAS OLAP Cube Studio and SAS Management Console as follows:
  - Use SAS Management Console to define, in the metadata, the server that will be used to access the tables. This is a SAS application server with a workspace server component.
  - Use SAS Management Console to define, in the metadata, the SAS library that contains the table.
  - In SAS OLAP Cube Studio, specify the server that will be used to access the tables. To set the server, select **Tools ▶ Options**Or, if the shortcut bar is displayed, select **`Options`** to set the server.
  - In SAS OLAP Cube Studio, select **Source Designer** to load the table definitions (or other information source) as follows:
    - From the shortcut bar, select **Tools ▶ Source Designer** or select **Source Designer**
    - Select a Source Type (SAS, ODBC, etc.), and then select **`Next`**.
    - If you have not specified a server, or if the server that is specified is not valid, then you will be prompted again for a server.
    - Select the SAS Library that contains the tables that you want to define, and then select **`Next`**.
    - Select the tables to define, and then select **`Next`**.
    - Select **`Finish`**. The table definitions are loaded into the metadata.
  - If you start to create a cube and do not see the table that you need to continue, then you can select the **`Define Table`** button in any of the windows that prompt for tables.
- In the Finish window of the cube designer, you are given the option to create the physical cube. The metadata definition is always stored as you leave the Finish window. However, you can defer creation of the physical cube. If you choose to create the cube as you leave the Finish window, then you must have a SAS Workspace Server defined that you can submit PROC OLAP code to. This server is defined in SAS Management Console.

Note: For further information about the different data types that you can use to load cubes from, see “Loading Cubes” on page 109. △

Note: The SAS Metadata Server enables duplicate librefs to be defined in the metadata. To ensure that the correct SAS library definition is found on the metadata server, you should assign the libref by using the LIBNAME statement for the metadata engine before submitting the PROC OLAP code. Otherwise, PROC OLAP will select the first library definition that it finds with your specified libref, and it will associate your

cube metadata with that definition. The selected library definition might or might not contain a description of the SAS data set that was actually used to build your cube. For more information about using the LIBNAME statement for the metadata engine, see "Statements" in *SAS Language Reference: Dictionary*. △

Note: When a SAS OLAP cube is created, a directory for that cube is also created. This directory is assigned the same name as the cube, but in uppercase letters. For example: When you save a cube in c:\olapcubes and name the cube Campaigns, the cube is saved in the directory c:\olapcubes\CAMPAIGNS. △

For further information about preliminary setup and configuration steps see the *SAS Intelligence Platform: System Administration Guide* and the *SAS Intelligence Platform: Data Administration Guide*.

Storage Location Requirements for Cube Metadata and Related Objects

When storing metadata that describes a cube, the metadata objects that describe the cube and the cube's associated libraries and source tables must be stored in the same repository, or the metadata that describes the cube must be in a custom repository that is dependent on the repository that contains the library and table objects. Otherwise, you will not be able to create the cube. In addition, the library and table objects that are referenced by a cube must always be in the same repository. The following options illustrate these conditions:

- The library, table, and cube objects can be in a Foundation repository.
- The library, table, and cube objects can be in Project A, which is dependent on the Foundation repository.
- The library and table objects can be in the Foundation repository, and the cube object can be in Project A.
- The cube object cannot be in the Foundation repository, and the library and table objects cannot be in Project A.
- The table object cannot be in the Foundation repository, and the library and cube objects cannot be in Project A.
- The library object cannot be in the Foundation repository, and the table and cube objects cannot be in Project A.

Defining Member Properties

When you create a SAS OLAP cube, the information that is relevant to the cube is defined with the cube hierarchy, measures, and aggregations (summaries) that will be stored with the cube. Additional information that is part of the cube member data can be included in the cube definition as a member property.

Member properties are attributes of dimension members that provide an additional gradation of information to users of the cube data. Member property information is usually not as significant as the levels and members within a dimension, and therefore, does not qualify as a level or member. However, it often has additional analytical value that can be useful at query time.

A member property is assigned to a level within a hierarchy, and a level can have multiple properties that are assigned to it. For hierarchy placement, a member property is assigned (by default) to all hierarchies that the select level is in. However, you can remove one or more (but not all) of the hierarchies that the member property is assigned to.

When you create a member property, you must specify the name, column, and level. Member property names can be shared across a cube but must be unique for a specific level within a specific hierarchy. You can also specify a caption, description, and format. The format that you specified here will be used instead of the format in the data set.

Property Statement

The PROPERTY statement is used with the PROC OLAP statement when you define a cube:

```
PROPERTY zipcode-region
   column=post_code
   hierarchy=geographic
   level=region;
```

Cube Designer

You can also establish member properties with the Member Property dialog box that is part of the Cube Designer interface in SAS OLAP Cube Studio. This is accessed after measures are defined, when you create a cube or edit a cube. Select **Add** to create a member property. At the Define a Member Property dialog box, enter the member property name, level, column, and caption.

Defining Distinct Count Measures

You can define distinct count statistics as measures with a cube using either the SAS OLAP Cube Studio, Define Distinct Count Measures function or the PROC OLAP MEASURE statement and the NUNIQUE statistic.

Define Distinct Count Measures

You can define a distinct count statistic on the Cube Designer wizard in SAS OLAP Cube Studio. On the Select Measures window, click the **Define Distinct Count** button. The Define Distinct Count Measures window opens. This window enables you to store NUNIQUE (Distinct Count) statistics as measures with a cube. You can add or delete a Distinct Count measure on this window. To create the Distinct Count measure, select the level and hierarchy for the new measure. Selecting the level in a tree structure (dimension, hierarchy, level) will automatically assign the dimension and hierarchy to that measure. A name will automatically be generated for the Distinct Count measure consisting of the level name, "NUNIQUE", and the parent hierarchy name. By default, Distinct Count measures will have the word "NUNIQUE" in their measure name.

Note: Only one level-hierarchy combination can be defined for a measure. After a level-hierarchy combination has been used to create a Distinct Count measure, the combination cannot be used again. △

Note: If the level or hierarchy is deselected, then any associated defined Distinct Count measures will be deleted from the cube. △

You can modify the name, caption, format, units, and description for a Distinct Count measure in the Cube Designer - Measure

Details window. Distinct Count measures are displayed on the Edit details for each measure table with the other selected measures. The Distinct Count measures are also listed on the **`Default Measure`** drop-down box and can be selected as a default measure.

NUNIQUE — You can define a distinct count statistic using the MEASURE statement and the NUNIQUE statistic. The LEVEL and HIERARCHY options for the MEASURE statement are used with the NUNIQUE statistic and will be ignored for non-NUNIQUE statistics if specified.

Note: The LEVEL name is optional. If it is omitted then the level name is assumed to be the name specified for the NUNIQUE measure. The HIERARCHY name is only required if the level is in multiple hierarchies. For further information about using the NUNIQUE statistic see the "MEASURE Statement" on page 95. △

Defining a Default Hierarchy

When you define cube dimensions, levels, and hierarchies in SAS OLAP Cube Studio, a default hierarchy for a dimension will automatically be created if a hierarchy is not explicitly defined. This default hierarchy includes all levels that were specified for the current dimension and the order they were listed in for the dimension. In addition, if you define multiple hierarchies and do not select a default, then the default is automatically assigned to the first hierarchy that is created for the dimension. On the Dimension Designer — Hierarchy window, you can click the **`Default`** button to set a selected hierarchy as the default for the dimension.

Defining Multiple Hierarchies for a Dimension

SAS OLAP cubes are organized into dimensions and levels of data. The levels are then arranged into hierarchies. After an initial hierarchy has been created, you can define additional hierarchies for a single dimension of a cube. This enables you to have multiple possible drill paths of the same data. When you create more than one hierarchy for a dimension, the levels have some restrictions:

- A level in a dimension might be used in more than one hierarchy within that dimension. However, levels cannot be used in hierarchies that are not defined within the dimension that the level is defined in.
- Each level must be used in at least one hierarchy.
- Levels from the same dimension that are picked for an aggregation must be in the drill order for at least one hierarchy in that dimension.
- You cannot share levels between dimensions.

You can arrange the levels in a hierarchy in any order. The one exception to this is the Time dimension. Levels in hierarchies in the Time dimension must follow a prescribed order that is determined by the numerical value that is assigned to the type. This order is from the smallest value (Years, 16) to the greatest value (Seconds, 3,096). You can only have one time dimension for a cube. The dimension hierarchies also have some restrictions:

- The first hierarchy that is defined for the dimension is designated as the default. When there are multiple hierarchies, you can designate the default hierarchy for the dimension.

- Hierarchy names must be unique across the cube. If there is a single hierarchy for a dimension, then its name must be the name of the dimension. Also, dimension and hierarchy names cannot be the same as a level name within that dimension.
- For any cube loaded with a star schema, in which a dimension table represents multiple hierarchies for that dimension, the dimension key that is used to join the dimension table to the fact table will be used for all hierarchies of that dimension.

Hierarchies Statement

The HIERARCHY statement is used with the PROC OLAP statement when you define a cube:

```
hierarchy campaigns
   levels=(campaign_type campaign sub_campaign);
```

Cube Designer

You can establish multiple hierarchies by using the Cube Designer - Dimensions window, which is located in the SAS OLAP Cube Studio Cube Designer. To add a hierarchy to an existing dimension, select a dimension, and then click **Modify**. This opens the Dimension Designer - General window. It is populated with the values for the selected dimension. Select **Next** until you reach the Dimension Designer - Hierarchy window. Select **Add** to create an additional hierarchy.

Note: You can modify existing hierarchies by selecting a hierarchy and clicking **Modify**. You can also assign a default hierarchy by selecting a hierarchy and clicking **Default**. The first hierarchy is automatically the default hierarchy. △

Note: An exception to defining multiple hierarchies for a dimension is the Time dimension. Levels in hierarchies in the Time dimension must follow a prescribed order that is determined by the numeric value that is assigned to the type. This order is from the smallest value (Year, 16) to the greatest value (Seconds, 3,096). △

In the Dimension Designer - Define a Hierarchy window, you can define a new hierarchy and select the different levels and their order for the hierarchy.

Defining Ragged and Unbalanced Hierarchies for a Dimension

Dimension levels are arranged in one or more hierarchies. Hierarchies, by process of ordering, have a branching arrangement, and the different member levels have parent and child relationships. For instance, at company X the sales staff are located in different regions and cities in different countries. A balanced hierarchy might look like this:

- Global sales president (top of hierarchy)
- Sales presidents (per country)
- Regional sales managers
- City sales managers.

Because of differences in the cube data, hierarchies are often not balanced and possibly have missing members. For example, some sales regions might not have sales managers assigned to a specific city. Or, some countries might not have sales regions, just cities. These real-world scenarios would create hierarchies that have missing member data and possibly ragged hierarchies. This affects the drillpath of the cube data.

You can also drill to missing members within a path and continue to drill down to members that are present.

Defining Ragged and Unbalanced Hierarchies in SAS OLAP Cube Studio

The Cube Designer in SAS OLAP Cube Studio enables you to specify the missing members for a hierarchy and the type of data that is missing. Here are the Cube Designer windows that enable you to specify missing member information:

Ragged Hierarchies

Located in the Cube Designer General - Advanced Cube Options window, this tab enables you to specify character and numeric missing member information. By default, no missing member information is indicated with the value **`None`**.

Dimension Designer - Level Properties

- **`Ragged - Ignore Missing Members`** specifies whether to ignore or use global or hierarchy-specific ragged hierarchy settings. To ignore settings, set this property to **`True`**. To use the settings, set this property to **`False`**. By default, this is set to **`False`**.
- **`Ragged - Designate Missing Members`** specifies that the Cube Designer use the specified string to identify missing values and override any global or hierarchy-specific ragged hierarchy settings. You can use up to 256 characters. The value of the True/False setting in Ragged - Ignore Missing Members controls whether or not you override any global or hierarchy-specific ragged hierarchy settings.

Dimension Designer - Define a Hierarchy

You can select one of these options from the Ragged Hierarchies tab:

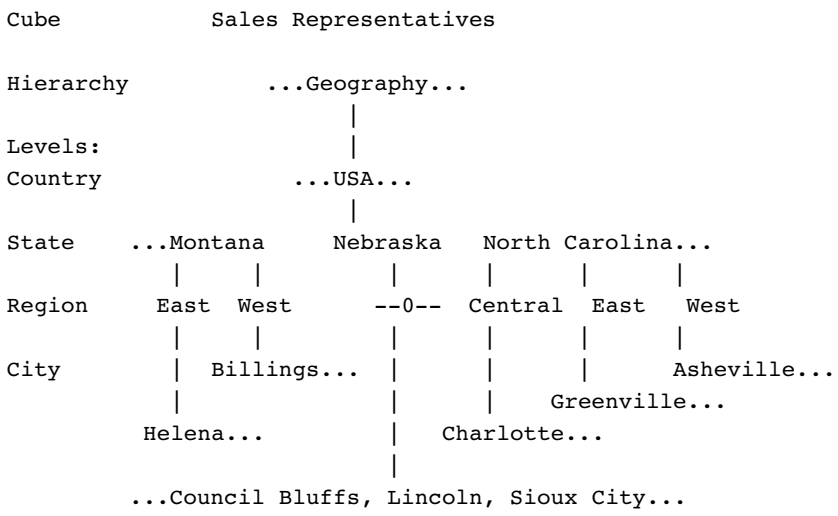
- **`Ignore`**. From this drop-down list, select **`True`** to ignore the global missing member settings that you entered at the Advanced Cube Options dialog box. Select **`False`** to use the global settings for the current hierarchy.
- **`Character`**. For this hierarchy only, enter a maximum of 256 characters that will be used to identify missing character members.
- **`Numeric`**. For this hierarchy only, enter a maximum of 256 characters that will be used to identify missing numeric members.

Defining Ragged and Unbalanced Hierarchies with PROC OLAP

To create ragged and unbalanced hierarchies with PROC OLAP, you specify options that allow the procedure to skip over members of levels that have captions with specified values. The presence of these skipped members constitutes a ragged or unbalanced hierarchy. In a ragged hierarchy, skipped members in a given level can have descendants; skipped members are used to enable drill-down through empty levels. In an unbalanced hierarchy, the skipped members do not have descendants; members are skipped in order to create hierarchies where certain branches do extend to all available levels.

For an example of the creation of a ragged hierarchy, assume that a cube defines information about sales representatives. The Geography hierarchy is defined to have the levels Country, State, Region, and City. In this particular sales organization, the

state of Nebraska has no regions, but it does have sales representatives in a number of cities. This ragged hierarchy can be shown as follows:

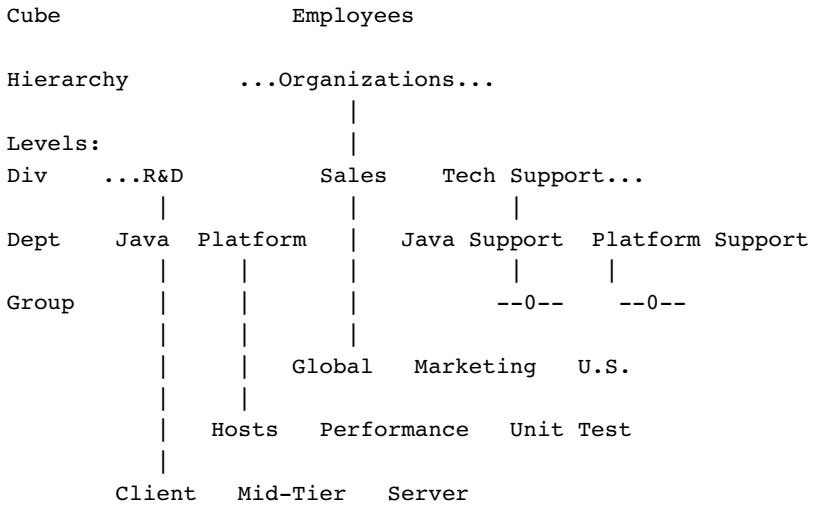


The Geography hierarchy is ragged because of the need to skip the Region level, and because the skipped member (Nebraska) has descendants.

To create the ragged hierarchy shown above, the Nebraska member needs to be defined with one member at the Region level. That member needs to have a caption that matches the value of the EMPTY_CHAR= option that is defined in the respective HIERARCHY statement (for option details, see "The OLAP Procedure" on page 78).

In the resulting cube, drilling down from Nebraska takes you directly from the State level to the City level.

For an example of the creation of an unbalanced hierarchy, assume that a cube named Employees has a hierarchy named Organizations. In that hierarchy there are various divisions, departments, and groups. As show below, some departments lack groups:



The preceding hierarchy is unbalanced because the Tech Support level has no descendants at the Group level. To implement this unbalanced hierarchy, the levels Java Support and Platform Support would have to be defined with captions that matched the value of the EMPTY= option in their respective LEVEL statements.

The options that implement ragged and unbalanced hierarchies are found in the PROC OLAP statement, HIERARCHY statement, and LEVEL statement. In the PROC OLAP and HIERARCHY statements you can specify separate caption values for

character and numeric levels using the options EMPTY_CHAR= and EMPTY_NUM=. Similarly, the EMPTY= option of the LEVEL statement allows you to specify separate values for each level in a hierarchy, regardless of any similar values that were specified in preceding HIERARCHY and PROC OLAP statements.

The HIERARCHY and LEVEL statements also provide the IGNORE_EMPTY option, which specifies that any prior specifications of EMPTY_CHAR= or EMPTY_NUM= are to be ignored for that hierarchy or level.

Ragged Hierarchies and Unique Member Names

In a ragged hierarchy, the parent of a member might not be at the level directly above that member. Furthermore, not all children of a member are necessarily at the same level. This can lead to a situation where two children have the same unique name.

For example, in a geography hierarchy you might have the levels state, county, and city. The state Washington might have a child at the county level called *Olympia* and another child at the city level, also named *Olympia*. The city member is not a descendant of the county member of the same name. It is a child of Washington.

In a ragged hierarchy, levels can have an unconventional structure, and unpopulated levels are not assigned a token or placeholder. As a result, the unique name for the county member is **`Geography.[All Geography].Washington.Olympia`**, and the unique name for the city member is **`Geography.[All Geography].Washington.Olympia`**.

The result of this anomaly is that the city member cannot be asked for by a unique name in a query, either through MDX or an OLE DB for OLAP (ODBO) request for metadata. It will be returned in any set that contains it so the data that is associated with it is not lost. The same applies to the children of a member such as *Olympia*. Because the server searches through the hierarchy to validate member names, a request by name for a child of *Olympia* the city will result in a bad member name error. This is because the server actually searches under the county *Olympia*.

This situation occurs only when two members with the same name share a parent. Any number of *Olympia(s)*could exist under other parents with no unusual results.

Cube Design-Aggregations

When determining how to efficiently deliver the data in a multidimensional cube, fast query response is extremely important. This is accomplished by storing summarized data. Any combination of dimension levels can become a stored aggregation. Which aggregations are being stored has a direct effect on the SAS OLAP Server CPU usage, file I/O, and query response times. The aggregations that are being stored also affect cube build time and the absolute cube file size. Therefore, it is a trade-off between a single instance of resource use at cube build time and multiple instances of resource use at cube query time.

The aggregated data values for SAS OLAP cubes can be stored either with the cube in the cube's internal format or external to the cube in relational summary tables.

MOLAP	MOLAP aggregation storage is the cube-internal storage for aggregations. MOLAP aggregation tables are created as part of the cube creation step.
ROLAP	ROLAP aggregation storage is the cube-external summary tables. ROLAP summary tables need to be pre-calculated from the input data (using tools such as SQL or PROC MEANS/PROC SUMMARY) and made known to the cube at cube creation time.

MOLAP Aggregation Storage

SAS MOLAP aggregation storage maintains the cube data in the same table format as the format that is used by the SAS Scalable Performance Data (SPD) Engine. MOLAP aggregation storage takes advantage of key contraction and allows data access by using the cube's internal data representation directly.

MOLAP aggregation storage of SAS OLAP cubes has the same threading and scalability features as the files used by the SAS SPD Engine. The data and the index section of the files are stored in different physical files. This enables parallel access to the data and index sections. The data and index files themselves are stored in partitions, enabling parallel data retrieval within the same file. The partitions of the data and the index section can be distributed over multiple disc controllers, thus adding a further boost to the threaded and partitioned architecture by reducing contention and possible bottlenecks in the physical I/O.

ROLAP Aggregation Storage

ROLAP tables used in SAS OLAP cubes can be SAS data sets, SAS data views, or any tables or views accessible through a SAS engine. This extends the choice of available storage options for SAS OLAP cubes to include SPDE, SPDS, and any RDBMS product for which a SAS/ACCESS software product is available. ROLAP aggregation tables must conform to the structure of the input data. The columns that feed the dimension levels must have the same column names and attributes that were used in the input data when loading the cube. In addition, all aggregations must be stored in fully de-normalized form. Here are some guidelines to make aggregation columns for measures available:

- Each ROLAP aggregation table must include all columns for the cube's measures with stored statistics.
- SAS OLAP cube aggregations store the following statistics: SUM, N, NMISS, USS, MIN, and MAX. Other available statistics are derived from the stored statistics by internal calculations. For example, in order to include a measure for the AVG statistic in your cube, you need to make columns available in your ROLAP aggregation tables that were generated by using SUM and N (count).

ROLAP data requests can also run against the data that was used to create and load the cube, whether from a detail table or a star schema. The cube's input data can be used in place of the aggregation with the combination of the lowest level of all dimensions (often called NWAY or NWAY aggregation, which is a name borrowed from PROC MEANS/PROC SUMMARY where it denotes the combination of all CLASS variables).

Choosing MOLAP or ROLAP Aggregation Storage

MOLAP aggregation storage is optimized for SAS OLAP Server internal processing and has a minimal data-size footprint. It uses threaded, parallel data access and is tunable to any hardware environment. MOLAP aggregation storage is convenient because it doesn't require additional data management steps.

ROLAP aggregation storage enables you to use existing ROLAP schemas and reuse legacy SAS OLAP Server SAS 8 HOLAP structures. ROLAP aggregation storage enables users to use database systems and data servers of their choice to store and serve cube aggregation data. Existing processes can be used to create and access aggregation data to off-load and distribute data access, I/O, and rollup to server systems of the user's choice.

A hybrid approach is possible. For example, users with existing ROLAP structures can build a "light" SAS OLAP cube with no additional stored aggregations and add MOLAP aggregations to further tune the cube performance.

SAS OLAP Cube Size Specifications

When creating SAS OLAP cubes there are some size specifications for the various components of the cube.

Dimensions and hierarchies
: A cube can have a maximum of 128 hierarchies. The number of dimensions is 128 or less, if multiple hierarchies per dimension are used. There is no limit to the number of hierarchies per dimension.

Levels
: The maximum number of levels for a cube is 256. There can be up to 19 levels per hierarchy.

Members
: There is no limit to the number of members per cube. An individual hierarchy has an upper limit on the number of members. This limit is determined by the following formula:

```
log2(numer members in first level of hier)+log2(number members
 in second level of hier)+.....+log2(number members in nth level
of hier) must be < 64
```

Measures
: The maximum number of measures per cube is 1024.

Number of cubes per schema
: Whereas there are no absolute restrictions for the number of cubes per OLAP Schema, assigning a large number of cubes to a schema should be avoided. This is because the cubes compete for the OLAP server's cube cache and data cache at the time they are accessed and handled by the server.

Properties
: The maximum number of properties per level is 256.

CHAPTER

3

Cube Building Examples

Building a Cube from a Detail Table

SAS OLAP Cube Studio

You can build an OLAP cube by using the Cube Designer in SAS OLAP Cube Studio. In this example, we are using data from a recent product marketing campaign. We want to establish measures and summaries of various aspects of our data such as geographic location of potential customers, age of potential customers, and revenue summaries. Our data is held in a detail table called olapsio.campnrml.

1 Define the metadata profile. The File menu options for New Metadata Profile and Open Metadata Profile allow you to define and connect to a metadata server. At the Open a Metadata Profile dialog box, you can choose to either create a new metadata profile or edit an existing one. Enter the machine information of the metadata server that you will connect to and retrieve a data source from.
   - At the Connection Information dialog box, enter the machine ID, port, your user ID, and password.

     Note: These options are the equivalent of the METASVR statement options:
     - HOST=
     - PORT=
     - USERID=
     - PW=.

     △
   - At the Repository Selection dialog box, select a default repository.

2 Enter general cube information.
   - After you have established a metadata server, you can begin to create a cube. Select **`Cube Designer`** from the shortcut menu. At the Cube Designer –

General dialog box, enter the general cube information. For input type, you select **Detail Table**.

- Cube Name
- Description
- Repository
- OLAP Schema
- Path in the file system to store the cube
- Input Type

Note: These options are equivalent to the PROC OLAP and METASVR statement options:

- CUBE=
- DESC=
- REPOSITORY=
- OLAP_SCHEMA=
- PATH=.

△

- At the Cube Designer – Input dialog box, select a data source or detail table for your cube. If one does not exist for your data, select **Define Table**, and then define the source that you will import your metadata from.

  Note: These options are equivalent to the

  - Source Designer function in the Tools Menu
  - DATA|FACT= option for PROC OLAP.

  △

- At the Cube Designer - Drill-Through dialog box, you select or define an optional drill-through table. Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source.

  If a drill-through table does not exist for your data, select **Define Table**, and then define the source that you will import your metadata from.

Note: These options are equivalent to the

- Source Designer function in the Tools Menu
- DRILLTHROUGH_TABLE | DT_TABLE | DT_TBL= option for PROC OLAP.

△

The **Table Options** button that is available at both the Cube Designer - Input and the Cube Designer - Drill-Through dialog boxes, opens the **Table Options** dialog box. It enables you to specify data set options that are used to open the data set. For example, you could enter a WHERE clause or subsetting information that is then applied to the selected table when it is opened. The options are stored as part of the cube and then reapplied when the data is accessed at run time. You can also specify data set options in the Dimension Designer – General window (for use with star schemas) and the Stored Aggregates window (for use with summarized tables). For more information, see “Data Set Options” in *SAS Language Reference: Concepts*.

3 Define dimensions, levels, and hierarchies. Now that your basic metadata server and cube information has been entered, you can define the different dimensions and their respective levels and hierarchies. For this example, you will define the following dimensions and levels:

- Campaigns
  - campaign type
  - campaign
  - subcampaign.
- Campaign Dates
  - campaign start year
  - campaign start month
  - campaign start day.
- Geographic
  - division
  - region
  - client_id.
- Customer Age
  - age group1
  - age group2.
- Products
  - product group
  - product type.

Define the dimensions for the cube. For each dimension you will define the dimension, its levels, and its hierarchies.

- At the Cube Designer - Dimensions dialog box, select the **Add** button. This opens the Dimension Designer - General dialog box. Enter the following information:
  - dimension name
  - caption
  - description
  - type of dimension (standard or time)
  - sort order.

  Note: If you are using a star schema, enter the Dimension Tables Definition information. △
- Select **Dimension Levels** from the Dimension Designer - Levels dialog box.
- Next, define properties such as format, time type, and sort order from the Dimension Designer - Level Properties dialog box.
- Next, define hierarchies for the levels from the Dimension Designer - Define a Hierarchy dialog box.
- Repeat this process for each dimension.

Note: Use the DIMENSION, HIERARCHY, and LEVEL statements here. For time-specific levels in a dimension, the LEVEL statement is required. Also, there can be only one time-specific dimension and one GEO-specific dimension per cube. △

4 Define measures. You can now define the measures for the cube. In your example, you want measures for revenue totals and the number of customers you tracked during the campaigns. Define the measures for the cube at the Cube Designer - Select Measures dialog box.

   Modify any measure attributes such as Measure Captions and Formats at the Cube Designer - Measure Details dialog box.

   Note: The MEASURE statement is used here. △

5 Define member properties. You can now define the member properties for any needed cube members. A *member property* is an attribute of a dimension member. A member property is also an optional cube feature that is created in a dimension to provide users with additional information about members. For this example, you can define the zip code or postal code as a member property. Define member properties at the Cube Designer - Member Property dialog box.

   At the Define a Member Property dialog box, enter the member property name, level, column, and caption.

   Note: The PROPERTY= statement is used here. △

6 Define aggregations. You can now define the aggregations for the cube. *Aggregations* are summaries of detailed data that are stored with a cube or referred to by a cube. They can help contribute to faster query response. Define the aggregations for the cube from the Cube Designer - Generated Aggregations dialog box.

   Select the levels for the aggregation with the Specify a generated user-defined aggregation dialog box.

   Note: The AGGREGATION= statement is used here. △

7 Build the cube. You can now build the cube. You can choose to build the cube and register it to the metadata repository, or you can register the cube to the metadata repository. At the Cube Designer - Finish dialog box, review the settings for the cube, and then select one of the cube creations options.

   Select whether to save the generated PROC OLAP code. At the Save PROC OLAP Code dialog box, enter the file location where you want to save the resulting code.

   Select the **`Finish`** button from the Cube Designer - Finish dialog box.

Note: When a SAS OLAP cube is created, a directory for that cube is also created. This directory is assigned the same name as the cube, but in uppercase letters. For example, when you save a cube in c:\olapcubes and name the cube Campaigns, the cube is saved in the directory c:\olapcubes\CAMPAIGNS. △

Saving a Cube's PROC OLAP Code

In SAS OLAP Cube Studio, you can elect to save the PROC OLAP code that is generated when a cube is built. The code is saved to a text file that you specify. The information saved in the file includes the following items:

- the SAS LIBNAME statement
- any FMTSEARCH statements
- any additional SAS code
- the PROC OLAP statement
- the METASVR statement
- all other PROC OLAP statements.

You can access the Save PROC OLAP Code window by using one of the following methods:

1. On the navigation tree in SAS OLAP Cube Studio, right-click on a cube and select **Save PROC OLAP code**.
2. In the Finish window in the Cube Designer wizard, right-click the **Save PROC OLAP Code** button.

The Save PROC OLAP Code window opens. In the Path field, enter the name and location of the file that you are saving to. An example is c:/temp/olapcode.txt.

PROC OLAP

You can build an OLAP cube with PROC OLAP and execute it in a SAS session. Running PROC OLAP registers your cube and its sources in a metadata repository. It also creates the files that make up the cube. These are the *possible* input types for an OLAP cube:

- a fact table (specified in the PROC OLAP statement DATA= option)
- dimension tables (specified in the DIMENSION statement DIMTBL= option)
- presummarized tables (specified in the AGGREGATION statement TABLE= option)

Note: You use the DATA= option when you use a detail table as the input. △

In this example, you are using data from a recent product marketing campaign. You want to establish measures and summaries of various aspects of your data such as geographic location of potential customers, age of potential customers, and revenue summaries. Our data is held in a table called olapsio.campnrml.

1. Define the metadata profile and general information. You use the PROC OLAP and METASVR statements here. The fact table is specified in the PROC OLAP statement DATA= option. The METASRV statement is used to establish the metadata connection. It identifies the metadata repository in which existing cube metadata information exists or in which metadata about a new cube should be stored. The statement is also used to provide a user's identification and password for the identified repository. Also the DRILLTHROUGH_TABLE= option is used here to indicate the drill-through table. Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source.

```
proc olap cube=Campaign1
   path="c:\cubes"
   drillthrough_table=olapsio.campnrml
   ;
metasvr host=localhost
   port=9999
   protocol=bridge
   userid=userid
   pw=pw
   repository=Foundation
   olap_schema="OLAP Schema"
   ;
```

   In OLAP Cube Studio, you define the above options in the following windows:

   - Metadata Profile
   - Connection Information

- □ Cube Wizard - General
- □ Cube Wizard - Input.

2 Define dimensions, levels, and hierarchies. Now that your basic metadata server and cube information has been entered, you can define the different dimensions and their respective levels and hierarchies. For this example, you define the following dimensions and levels:

- □ Campaigns
  - □ campaign type
  - □ campaign
  - □ subcampaign.
- □ Campaign Dates
  - □ campaign start year
  - □ campaign start month
  - □ campaign start day
- □ Geographic
  - □ division
  - □ region
  - □ client_id.
- □ Customer Age
  - □ age group1
  - □ age group2.
- □ Products
  - □ product group
  - □ product type.

You use the DIMENSION, HIERARCHY, and LEVEL statements here.

Note: For time-specific levels in a dimension, the LEVEL statement is required. Also, there can be only one time-specific dimension. △

```
dimension campaigns
   hierarchies=(campaigns)
   caption="Campaigns"
   sort_order=ascformatted
   ;
hierarchy campaigns
   levels=(campaign_type campaign sub_campaign)
   ;
dimension campaign_dates
   hierarchies=(campaign_dates)
   caption="Campaign launch dates"
   type=time
   ;
hierarchy campaign_dates
   levels=(campaign_start_year
      campaign_start_month
      campaign_start)
   ;
```

```
level campaign_start_year
   type=year
   ;
level campaign_start_month
   type=months
   ;
level campaign_start
   type=days
   ;
dimension geographic
   hierarchies=(geographic)
   caption="Geographic"
   ;
hierarchy geographic
   levels=(division region client_id)
   ;
dimension customer_age
   hierarchies=(customer_age)
   caption="Customer age"
   ;
hierarchy customer_age
   levels=(age_group_1 age_group_2)
   ;
dimension products
   hierarchies=(products)
   caption="Products"
   ;
hierarchy products
   levels=(product_group product_type)
   ;
```

In OLAP Cube Studio, the above options are defined in the following windows:

- Dimension Designer - General
- Dimension Designer - Levels
- Dimension Designer - Level Properties
- Dimension Designer - Define a Hierarchy.

3 Define measures. You can now define the measures for the cube, both stored and derived. A *measure* is an input column and a roll-up rule (statistic). Only certain measures are physically stored. Other measures are derived from the stored measures at run time. In this example, you want measures for revenue totals and the number of customers you tracked during the campaigns.

You use the MEASURE statement here.

```
measure revenue_sum
   column=revenue
   stat=sum
   format=dollar15.2
   ;
measure number_of_customers_sum
   column=number_of_customers
   stat=sum
   format=12.0
   ;
```

In OLAP Cube Studio, the above options are defined in the following windows:

- Cube Designer - Select Measures
- Cube Designer - Measure Details.

4 Define member properties. You can now define the member properties for any needed cube members. A *member property* is an attribute of a dimension member. A member property is also an optional cube feature that is created in a dimension to provide users with additional information about members. For this example, you can define the zip code or postal code as a member property.

You use the PROPERTY statement here.

```
property zipcode-region
   column=post_code
   hierarchy=geographic
   level=region
   ;
```

In OLAP Cube Studio, the above options are defined in the following windows:

- Cube Designer
- Member Property
- Define a Member Property.

5 Define aggregations. You can now define the aggregations for the cube. Aggregations are summaries of detailed data that is stored with a cube or referred by a cube. Their existence can reduce cube query time. If all aggregations are to be generated at the time of cube creation (MOLAP cube), then you can select specific aggregations that must be created in addition to the NWAY, which is the only aggregation that PROC OLAP makes by default.

You use the AGGREGATION statement here.

```
aggregation product_group product_type
   age_group_1 age_group_2
   division region
   ;
```

Note: The input type – presummarized tables – can also be specified in the AGGREGATION statement by using the TABLE= option. △

In OLAP Cube Studio, the above options are defined in the following windows:

- Cube Designer - Generate Aggregations
- Specify a generated user-defined aggregation.

6 Build the cube. You can now build the cube. Execute the PROC OLAP statement within the SAS System or in batch-mode. Here is the complete PROC OLAP code.

```
proc olap data=olapsio.campnrml
   cube=Campaign1
   path="c:\cubes"
   ;
metasvr host=localhost
   port=9999
   protocol=bridge
   userid=userid
   pw=pw
   repository=Foundation
   olap_schema="OLAP Schema"
   ;
```

```
dimension campaigns
   hierarchies=(campaigns)
   caption="Campaigns"
   sort_order=ascformatted
   ;
hierarchy campaigns
   levels=(campaign_type campaign sub_campaign)
   ;
dimension campaign_dates
   hierarchies=(campaign_dates)
   caption="Campaign launch dates"
   type=time
   ;
hierarchy campaign_dates
   levels=(campaign_start_year
      campaign_start_month
      campaign_start)
   ;
level campaign_start_year
   type=year
   ;
level campaign_start_month
   type=months
   ;
level campaign_start
   type=days
   ;
dimension geographic
   hierarchies=(geographic)
   caption="Geographic"
   ;
hierarchy geographic
   levels=(division region client_id)
   ;
dimension customer_age
   hierarchies=(customer_age)
   caption="Customer age"
   ;
hierarchy customer_age
   levels=(age_group_1 age_group_2)
   ;
dimension products
   hierarchies=(products)
   caption="Products";
hierarchy products
   levels=(product_group product_type)
   ;
measure revenue_sum
   column=revenue
   stat=sum
   format=dollar15.2;
measure number_of_customers_sum
   column=number_of_customers
   stat=sum
```

```
   format=12.0
   ;
aggregation product_group product_type
   age_group_1 age_group_2
   division region
   ;
run;
```

Note: Any libraries must be specified prior to running the PROC OLAP code. △

Note: When a SAS OLAP cube is created, a directory for that cube is also created. This directory is assigned the same name as the cube, but in uppercase letters. For example, when you save a cube in c:\olapcubes and name the cube Campaigns, the cube is saved in the directory c:\olapcubes\CAMPAIGNS. △

Building a Cube from a Summary Table

In this example, you can build a cube with fully summarized data. A *summary table* is a data source that contains a crossing of all dimensions for a cube. In this example, the data for the Campaign cube has been summarized into a table called CAMPAIGN_SUMMARY. The table contains a column

- for each of the levels that you want
- for each of the stored measures, total revenue, and total number of customers
- for the member property and zip code region

The table contains only the NWAY. The data set was produced with the following SAS code:

```
proc summary data=olaplib.campaign
   nway
   noprint
   ;
class campaign_type campaign
   sub_campaign campaign_start_year
   campaign_start_month campaign_start
   division region client_id
   age_group_1 age_group_2
   product_group product_type
   ;
var revenue number_of_customers
   ;
id post_code
   ;
output out=olaplib.campaign_summary
   sum=totrev totcust
   ;
run;
```

To create the cube by using SAS Cube Studio, follow these steps:

1. Start SAS Cube Studio and connect to the appropriate metadata server.
2. Define the campaign_summary table in the metadata by using the Source Designer.
3. To create a cube, select **Cube Designer** from the shortcut menu.

4 At the Cube Designer - General dialog box, enter the following information:
   - Cube Name
   - Description
   - Repository
   - OLAP Schema
   - Path in file system to store the cube
   - Input Type.

   For input type, select **Fully Summarized Table**.

5 At the Cube Designer - Input dialog box, select a data source or detail table for your cube. For this example, select the **campaign_summary** table.

   Note: If a detail table does not exist for your data, then select **Define Table** and define the source that you will import your metadata from. △

   Note: When you create a cube from a detail table or star schema, it is the equivalent of specifying the DATA= or FACT= options in the PROC OLAP statement. When you select Fully Summarized Table, this is the same as not specifying DATA= or FACT= and then specifying an AGGREGATION statement that has all levels listed and has the TABLE= option specified with the table name. Here is an example:

```
aggregation campaign campaign_type /
   table=olaplib.campaign_summary
   name='Default'
   ;
```

   △

6 On the Cube Designer - Drill-Through panel, select a drill-through table. When selecting a drill-through table (and you use a fully summarized table to load the cube) you should select the table that was used to create the summarized table. In this example, that is the campaign table. Select the radio button **Select drill-through table from list**, and then select the **Campaign** table.

7 Now that your basic metadata server and cube information has been entered, you can define the different dimensions and their respective levels and hierarchies. This example cube has these dimensions:
   - Campaigns
     - campaign type
     - campaign
     - subcampaign.
   - Campaign Dates
     - campaign start year
     - campaign start month
     - campaign start day.
   - Geographic
     - division
     - region
     - client_id.
   - Customer Age
     - age group1
     - age group2.

- Products
  - product group
  - product type.

Define the dimensions for the cube. For each dimension, define the dimension, its levels, and its hierarchies.

- At the Cube Designer - Dimensions dialog box, select the **Add** button. This opens the Dimension Designer - General dialog box. Enter the following information:
  - dimension name
  - caption
  - description
  - type of dimension (standard or time)
  - sort order.
- Select the necessary dimension levels at the Dimension Designer - Levels dialog box.
- Define properties such as format, time type, and sort order at the Dimension Designer - Level Properties dialog box.
- Define hierarchies for the levels at the Dimension Designer - Define a Hierarchy dialog box.
- Repeat this process for each dimension.

Note: You use the DIMENSION, HIERARCHY, and LEVEL statements here. For time-specific levels in a dimension, the LEVEL statement is required. Also, there can only be one time-specific dimension and one GEO-specific dimension per cube. △

8 You can now select the stored (base) measures for the cube in the Cube Designer - Select Stored Measures window. When loading from a detail table, the base and derived measures are generated from a single column in the detail table. For example, a detail table that has a column ACTUAL can have two measures – ACTUAL_SUM and ACTUAL_AVG – that are created from the column. However, with a fully summarized table, you must have one column for any base measure that you want to include in the cube. The base statistics are SUM, N, NMISS, MIN, MAX, and USS. Measures that are created with these statistics must have a single column in the summarized table. For this example, we have two base or stored measures: TOTREV and TOTCUST. Select the two columns in this window.

Note: This step is equivalent to using the AGGR_COLUMN option in the MEASURE statement. △

9 In the Cube Designer - Assign Stored Measures window, you can specify the Statistic and Analysis Group options for the stored statistics. Select **SUM** as the statistic for both the TOTREV and TOTCUST measures. For Analysis group, specify REVENUE for TOTREV and NUMBER_OF_CUSTOMERS for TOTCUST. If the table contained two measures from the same analysis column, both of the base measures would have the same analysis group specified. For example, if campaign_summary contained a column called REVN, which is the number of non-missing values for the REVENUE column, then we could have a base measure REVN with the statistic of count (N) and an analysis group of REVENUE.

Note: The analysis group is equivalent to the COLUMN|ANALYSIS option in the MEASURE statement. △

10 In the Cube Designer – Select Derived Measures window, specify any measures that will be derived from the base or stored measures. Each derived measure is

based on a set of required stored measures. If the stored measures for an analysis group do not include all those required for a specific derived measure, then that measure cannot be included in the cube. For example, TOTREV is the SUM of the REVENUE group. You cannot include AVGREV because you do not have the N or count of the REVENUE group in the stored measures.

11 In the Cube Designer - Edit Measure Details window, specify captions, formats, and other information about the measures that are listed.

12 Define the member properties for any needed cube members. A member property is an attribute of a dimension member. A member property is also an optional cube feature that is created in a dimension to provide users with additional information about members. Define member properties at the Cube Designer - Member Property dialog box.

At the Define a Member Property dialog box, enter the member property name, level, column, and caption.

Note: You use the PROPERTY= statement here. △

13 Define the aggregations for the cube. Aggregations are summaries of detailed data that is stored with a cube or referred by a cube. They can contribute to faster query response. If you have additional aggregation tables, select them in the Cube Designer - Aggregation Tables window.

14 Define the stored aggregations in the Cube Designer - Stored Aggregations window. These are aggregations that are contained in the additional input tables that were selected in the Cube Designer - Aggregation Tables window.

Note: Defining aggregations in this panel is equivalent to using the AGGREGATION statement with the TABLE= option. △

15 In the Cube Designer - Generated Aggregations window, define additional aggregations that will be generated when the cube is built.

16 Build the cube. In the Cube Designer - Finish window, you can select whether or not you want the cube to be physically created after the metadata is saved. When you select the **Finish** button, the metadata for the cube is always saved. If you select **Save the metadata and create the cube**, the short form of the PROC OLAP code is generated along with the necessary LIBNAME statements and submitted to an application server. You can also select whether to save the PROC OLAP code that is generated. At the Save PROC OLAP Code dialog box, enter the file location where you want to save the resulting code.

Note: When a SAS OLAP cube is created, a directory for that cube is also created. This directory is assigned the same name as the cube, but in uppercase letters. For example, when you save a cube in c:\olapcubes and name the cube Campaigns, the cube is saved in the directory c:\olapcubes\CAMPAIGNS. △

Here is the complete PROC OLAP code:

```
proc olap cube=Summary1
   drillthrough_table=olaplib.CAMPAIGN_SUMMARY
   path="c:\cubes"
   description="Summary1"
   ;
metasvr host=localhost
   port=9999
   protocol=bridge
   userid=userid pw=pw
   repository=Foundation
   olap_schema="OLAP Schema"
```

```
   ;
dimension Campaigns
   hierarchies=(Campaigns )
   caption='Campaigns'
   sort_order=ascending
   ;
hierarchy Campaigns
   levels=( Campaign_Type Campaign Sub_Campaign )
   caption='Campaigns'
   default
   ;
level Campaign_Type
   caption='Campaign Type'
   sort_order=ascending
   ;
level Campaign
   caption='Campaign ID'
   sort_order=ascending
   ;
level Sub_Campaign
   caption='Sub Campaign'
   sort_order=ascending
   ;
dimensioon CampaignDates
   hierarchies=(CampaignDates )
   caption='CampaignDates'
   type=time
   sort_order=ascending
   ;
hierarchy CampaignDates
   levels=( Campaign_Start_Year
      Campaign_Start_Month Campaign_Start )
   caption='CampaignDates1'
   default
   ;
level Campaign_Start_Year
   type=year
   caption='Campaign Start Year'
   sort_order=ascending
   ;
level Campaign_Start_Month
   type=months
   caption='Campaign Start Month'
   sort_order=ascending
   ;
level Campaign_Start
   type=days
   caption='Campaign Start'
   sort_order=ascending
   ;
dimension Geographic
   hierarchies=( Geographic )
   caption='Geographic'
   sort_order=ascending
```

```
   ;
hierarchy Geographic
   levels=( Division Region Client_ID )
   caption='Geographic'
   default
   ;
level Division
   caption='Division'
   sort_order=ascending
   ;
level Region
   caption='IFA Region'
   sort_order=ascending
   ;
level Client_ID
   caption='Client id'
   sort_order=ascending
   ;
dimension CustomerAge
   hierarchies=(CustomerAge )
   caption='CustomerAge'
   sort_order=ascending
   ;
hierarchy CustomerAge
   levels=( Age_Group_1 Age_Group_2 )
   caption='AgeGroup1'
   default
   ;
level Age_Group_1
   caption='Age Group 1'
   sort_order=ascending
   ;
level Age_Group_2
   caption='Age Group 2'
   sort_order=ascending
   ;
dimension Products
   hierarchies=( Products )
   caption='Products'
   sort_order=ascending
   ;
hierarchy Products
   levels=( Product_Group Product_Type )
   caption='Products1'
   default
   ;
level Product_Group
   caption='Product Group'
   sort_order=ascending
   ;
level Product_Type
   caption='Product Code'
   sort_order=ascending
   ;
```

```
measure totrevSUM
   stat=sum
   analysis=Revenue
   aggr_column=totrev
   caption='Sum of Revenue'
   format=BEST12.
   default
   ;
 measure totcustSUM
   stat=sum
   analysis=Number_Of_Customers
   aggr_column=totcust
   caption='Sum of Number_Of_Customers'
   format=BEST12.
   ;
run;
```

Building a Cube from a Star Schema

In this example, you can build a cube from a star schema. A *star schema* is a data source that contains tables in a database in which a single fact table is connected to multiple dimension tables. In this example, an international retail company sells sports and outdoor products.

To create the cube by using SAS Cube Studio, complete these steps:

1. Start SAS OLAP Cube Studio and connect to the appropriate metadata server.
2. To begin creating a cube, select **Cube Designer** from the shortcut menu.
3. In the Cube Designer - General window, enter the following information:
   - Cube Name
   - Description
   - Repository
   - OLAP Schema
   - Path in file system to store the cube
   - Input Type.

   For input type, select **Star Schema**.
4. In the Cube Designer - Input window, select a data source or detail table for your cube. For this example select the **ORDER_FACT** table. If a detail table does not exist for your data, select **Define Table**, and then define the source that you will import your metadata from.

   Note: When you build the dimensions using PROC OLAP, and if you use a fact table from a star schema, use the FACT= option. Additionally, use FACT= option to read a star schema, then use the DIMTBL= option to specify the dimension tables. △

   Note: If the cube is built from a star schema, then the keys that link the dimension table and the fact table are also defined by using the DIMKEY= and FACTKEY= options. See the "DIMENSION Statement" on page 86 for further information. △
5. In the Cube Designer - Drill-Through window, determine whether or not you will have a drill-through table. In this example, you will not use a drill-through table, so you can select the option **No table for Drill-Through**.

6 In the Cube Designer - Dimension Tables window, select dimension tables that are associated with the ORDER_FACT star schema that you specified as the data source for the cube. For this example, select the following tables:
   - CUSTOMER_DIM
   - GEOGRAPHY_DIM
   - ORGANIZATION_DIM_MOD_LEVELLED
   - TIME_DIM.

7 Now that your basic metadata server and cube information has been entered, define the different dimensions and their respective levels and hierarchies. This example cube has these dimensions and levels:
   - Time
     - Year_ID
     - Quarter
     - Month_Name
     - Week_Name
     - Date_ID.

     For the Time dimension, the following star schema information is also included:

Table	**TIME_DIM**
Key	Date_ID
Fact Key	Order_Date

   - Customers
     - Customer_Name
     - Customer_Age
     - Customer_Gender
     - Customer_Group
     - Customer_Type.

     For the Customers dimension, the following star schema information is also included:

Table	**CUSTOMER_DIM**
Key	Customer_Id
Fact Key	Customer_Id

   - Geography
     - Continent_Name
     - Country
     - State
     - Region
     - Province

- County
- City.

For the Geography dimension, the following star schema information is also included:

Table	GEOGRAPHY_DIM
Key	Street_Id
Fact Key	Street_Id

- Organization
  - Employee_Name
  - Job_Title
  - Salary
  - Gender
  - Company
  - Department
  - Org_Group
  - Section.

For the Organization dimension, the following star schema information is also included:

Table	ORGANIZATION_DIM
Key	Employee_Id
Fact Key	Employee_Id

Define the dimensions for the cube. For each dimension, you define the dimension, its levels, and its hierarchies.

- At the Cube Designer - Dimensions dialog box, select the **Add** button. This opens the Dimension Designer - General dialog box. Enter the following information:
  - dimension name
  - caption
  - description
  - type of dimension (standard or time)
  - sort order.

  When you define the dimensions for a cube based on a star schema, you will need to provide additional information about the dimensions in the Dimension Designer - General window. On the Star Schema Dimension Tables Definition panel, enter the following information:
  - Table
  - Key
  - Fact Key
  - Data Set Options.

- □ Select the necessary dimension levels at the Dimension Designer - Levels dialog box.
- □ Define properties such as format, time type, and sort order at the Dimension Designer - Level Properties dialog box.
- □ Define hierarchies for the levels at the Dimension Designer - Define a Hierarchy dialog box.
- □ Repeat this process for each dimension.

Note: You use the DIMENSION, HIERARCHY, and LEVEL statements here. For time-specific levels in a dimension, the LEVEL statement is required. Also, there can be only one time-specific dimension and one GEO-specific dimension per cube. △

8 Specify the columns or measures for the cube. In the Cube Designer - Selected Measures window, select the following columns and associated Sum statistics:

- □ Total_Retail_Price /Sum
- □ Quantity /Sum
- □ CostPrice_Per_Unit /Sum
- □ Discount /Sum.

9 Specify detail information for the measures. In the Cube Designer - Measure Details window, enter any necessary information for the different measures:

- □ Caption
- □ Format
- □ Unit
- □ Description.

For the measure Total_Retail_Price, enter a format value of DOLLAR12.2. For the measure CostPrice_Per_Unit, enter a format value of DOLLAR10.2.

10 Specify member property information for the levels in the cube. In the Cube Designer - Member Property window, select the **Add** button to create a new member property. In the Define a Member Property window, enter the following information about the member property:

- □ Name
- □ Level
- □ Column
- □ Format
- □ Caption Description
- □ Selected Hierarchies.

In this example, the following member properties are created:

Property Name	Level	Column	Caption	Selected Hierarchy
WeekDay_Number_US	date	weekday_no	US WeekDay Number	
WeekDay_Number_EU	date	weekday_eu	EU WeekDay Number	
Week_Number_EU	week_name	week_no	EU Week Number	YWD
Month_Number	month_name	month_no	Month Number	YMD

Property Name	Level	Column	Caption	Selected Hierarchy
Month_Number	month_name	month_no	Month Number	YQMD
Holiday_US	date	Holiday_US	US Holidays	

11 Specify the aggregations for the cube. Aggregations are summaries of detailed data that is stored with a cube or referred by a cube. They can contribute to faster query response. In the Cube Designer - Generated Aggregations window, select the **Add** button to specify aggregations and associated levels. Order the levels for the aggregations to follow the hierarchy drill path. The aggregations include
   - RegionalCustomerUse
   - QuarterlyCustomerUse
   - YearlyCustomerUse
   - WorldwideStaff
   - WorldwideSalaries.

   Note: When you create cubes in SAS OLAP Cube Studio, a default aggregation, which is the NWAY aggregation, is automatically created and listed in the Cube Designer - Generated Aggregations window. △

12 Build the cube. In the Cube Designer - Finish window, select whether or not you want the cube to be physically created after the metadata is saved. When you click **Finish**, the metadata for the cube is always saved. If you select **Save the metadata and create the cube**, the short form of the PROC OLAP code is generated along with the necessary LIBNAME statements and submitted to an application server. You can also select whether to save the PROC OLAP code that is generated. At the Save PROC OLAP Code dialog box, enter the file location where you want to save the resulting code.

Note: When a SAS OLAP cube is created, a directory for that cube is also created. This directory is assigned the same name as the cube, but in uppercase letters. For example, when you save a cube in c:\olapcubes and name the cube Campaigns, the cube is saved in the directory c:\olapcubes\CAMPAIGNS. △

Here is the complete PROC OLAP code:

```
proc olap cube=Star
   path="c:\cubes"
   fact=olapsio.ordfact
   ;
metasvr host=localhost
   port=9999
   protocol=bridge
   userid=userid
   pw=pw
   repository=Foundation
   olap_schema="OLAP Schema"
   ;
dimension Time
   hierarchies=(YWD YMD YQMD)
   type=time
```

```
   dimtbl=olapsio.timedim
   dimkey=date_ID
   factkey=order_date
   ;
hierarchy YWD
   caption="Year-Week-Day"
   levels=(Year_ID Week_Name Date_ID
   ;
hierarchy YMD
   caption="Year-Month-Day"
   levels=(Year_ID Month_Name Date_ID)
   ;
hierarchy YQMD
   caption="Year-Quarter-Month-Day"
   levels=(Year_ID Quarter Month_name Date_ID)
   ;
level year_ID
   type=year
   ;
level quarter
   type=quarters
   ;
level month_name
   type=months
   ;
level week_name
   type=weeks
   ;
level date_ID
   type=days
   ;
property WeekDay_Number_US
   caption="US WeekDay Number"
   column=weekday_no
   level=date
   ;
property WeekDay_Number_EU
   caption="EU WeekDay Number"
   column=weekday_eu
   level=date
   ;
property Week_Number_EU
   caption="EU Week Number"
   column=week_no
   hierarchy=YWD
   level=week_name
   ;
property Month_Number
   caption="Month Number"
   column=month_no
   hierarchy=YMD
   level=month_name
   ;
property Month_Number
```

```
   caption="Month Number"
   column=month_no
   hierarchy=YQMD
   level=month_name
   ;
property Holidays_US
   caption="US Holidays"
   column=Holiday_us
   level=date
   ;
dimension Customers
   hierarchies=(PersonalData CompanyUsage)
   dimtbl=olapsio.custdim
   dimkey=customer_id
   factkey=customer_id
   ;
hierarchy PersonalData
   levels=(Customer_Name Customer_Age Customer_Gender)
   ;
hierarchy CompanyUsage
   empty_char=_missing_
   levels=(Customer_Group Customer_Type)
   ;
dimension Geography
   hierarchies=(Geography)
   dimtbl=olapsio.geogdim
   dimkey=street_id
   factkey=street_id
   ;
hierarchy Geography
   empty_char=_missing_
   levels=(Continent_Name Country
      State Region Province
      County City)
   ;
dimension Organization
   hierarchies=(PersonalStats Organization)
   dimtbl=olapsio.orgdim
   dimkey=employee_id
   factkey=employee_id
   ;
 hierarchy PersonalStats
   levels=(Employee_name Job_Title Salary Gender)
   ;
 hierarchy Organization
   empty_char=_missing_
   levels=(Company Department
      Org_Group Section Job_Title)
   ;
measure DiscountSum
   stat=sum
   column=Discount
   ;
measure CostPrice_Per_UnitSum
```

```
   stat=sum
   column=CostPrice_Per_Unit
   format=DOLLAR10.2
   ;
measure QuantitySum
   column=Quantity
   caption='Sum of Quantity'
   ;
measure Total_Retail_PriceSum
   stat=sum
   column=Total_Retail_Price
   format=DOLLAR12.2
   ;
aggregation Continent_Name Country State
   Region Customer_Group Customer_Type
   / name='RegionalCustomerUse'
   ;
aggregation Year Quarter Customer_Group Customer_Type
   / name='QuarterlyCustomerUse'
   ;
aggregation Year Customer_Group Customer_Type
   / name='YearlyCustomerUse'
   ;
aggregation Continent_Name Country
   State Region Province Company
   Department Org_Group Section
   / name='WorldwideStaff'
   ;
aggregation Continent_Name Country State
   Region Province Employee_Name
   Job_Title Salary
   / name='WorldwideSalaries'
   ;
run;
```

CHAPTER

4

Modifying and Updating Cubes

Updating a Cube

A cube can be updated after its initial creation in order to optimize cube performance and to add or remove aggregations. Any cube update changes elements of both the physical cube and its metadata registration. You can update a cube in either SAS OLAP Cube Studio or with PROC OLAP code.

- To update an existing cube in SAS OLAP Cube Studio, select the cube that you want to modify, right-click, and select the **`Edit Cube Structure`** option. You can also access this function from the Actions menu. This opens the Cube Designer Wizard. Make any necessary changes to the cube. At the Cube Designer - Finish dialog box, review the settings for the cube and select one of the cube creations options. SAS OLAP Cube Studio deletes the cube, and then rebuilds it with the changes that you entered.
- To update a cube by using PROC OLAP code, modify the code as needed and save the updated code. Use the DELETE option, the DELETE_PHYSICAL option, or both from the PROC OLAP statement to delete the cube. Resubmit the modified code to rebuild the cube. See the DELETE and DELETE_PHYSICAL options for the "PROC OLAP Statement" on page 79 for further information. Here are three possible usage scenarios:

  If you are changing captions and descriptions or the dimensions or measures,
  You must use DELETE to remove the physical cube and the metadata registration. This is because you are submitting the full PROC OLAP syntax with changes.

  If the input table has new data and you only want to refresh the cube,
  You should use DELETE_PHYSICAL to delete only the physical cube. You can then submit the shorter form of the PROC OLAP syntax with only the PROC OLAP statement and METASVR statement.

  If you are optimizing cube performance by adding or deleting aggregations,
  You should not use the DELETE or DELETE_PHYSICAL options. This is because you are updating the cube in its current state.

Refreshing Cube Metadata

You can refresh the metadata for calculated members and named sets that are associated with a cube by using the Refresh Cubes function, which is available from the SAS OLAP Cube Studio toolbar. The refresh cubes function reads the information about calculated members and named sets that is stored on the metadata server. The refresh cubes function then updates the OLAP server metadata for the cube. You must be connected to a server and have administrative permissions in order to select the Refresh Cubes function.

With the Refresh Cubes function, you can select from a list of one or more cubes to refresh. You can also select a check box that selects all cubes. The cubes listed are

those cubes that are assigned to the current OLAP schema and that physically exist. When you have selected the cubes that you want to refresh, select **OK**. The refresh command is then sent to all cubes that were selected.

After the selected cubes have been refreshed, you are prompted to check other servers that the current OLAP schema is associated with.

MDX DDL REFRESH Statement

The REFRESH statement can be sent manually. You can send the REFRESH statement for each additional server that the schema is associated with.

```
REFRESH CUBE (cubename | "_ALL_" )
```

Where cubename specifies a single cube to refresh for the current server connection. Or _ALL_ specifies that all cubes are refreshed for the current server connection. Here are some examples.

This example uses the REFRESH statement to refresh the metadata associated with a cube named OrionStar.

```
refresh cube [OrionStar]
```

This example uses the REFRESH statement to refresh the metadata for all cubes managed by the connected server.

```
refresh cube _ALL_
```

You can use the OLAP MDX SQL Pass-Through facility to send the DLL REFRESH statement to a server. Here is an example.

```
proc sql noerrorstop;
  connect to olap (&olapcon);
  execute
  (
    refresh cube [OrionStar]
  ) by olap;

proc sql noerrorstop;
  connect to olap (&olapcon);
  execute
  (
    refresh cube _ALL_
  ) by olap;
```

Tuning Cube Aggregations

When a cube is created, aggregations can be specified by the user. Aggregations are usually created to improve query performance. After a cube is created and queries are run against the cube, users might discover that certain aggregations are not being used, and adjustments and changes to the aggregations are needed. You might want to change the levels in an aggregation, add another aggregation, or entirely remove an aggregation.

Tuning aggregations is important for improving query response times. After you have created and executed reports for your cubes, you can determine what are the most common queries generated or what are the least common queries generated. These can then be compared to the aggregations available. It is recommended to add those

aggregations that are commonly used (those that may only need a modification of an existing aggregation) and delete those that are not necessary.

There are three possible options to modify the aggregations for a cube:

- Manual Tuning (within SAS OLAP Cube Studio)
- Advanced Aggregation Tuning plug-in (within SAS OLAP Cube Studio)
- PROC OLAP

Using the Manual Tuning Function

The Manual Tuning function in SAS OLAP Cube Studio allows you to adjust and improve an existing cube by adding, dropping, or modifying aggregations. Manual Tuning requires an active IOM server connection. When you select Manual Tuning you are prompted for a SAS IOM Server. If no valid server is available, the Manual Tuning window will not open. In addition, only MOLAP cubes that have an existing physical cube, along with a SAS Metadata Repository registration, can use the Manual Tuning function.

You can access the Manual Tuning function within SAS OLAP Cube Studio by selecting an existing cube and right-clicking to display the menu that lists the Manual Tuning function. Or, you can also access it from the main OLAP Cube Studio menu under the Actions menu.

The Manual Tuning window lists the defined aggregations for the selected cube. When you finish making selections and select **OK**, the PROC OLAP statements are submitted to the IOM application server and the cube data is updated. With the Manual Tuning Window you can

- Add an aggregation - select **Add** to create a new aggregation. Enter the aggregation name and select the levels for the aggregation. Select **Apply** to save and validate the aggregation. If the aggregation is a duplicate or it is not in hierarchy order, you will receive an error message.
- Delete an aggregation - select an aggregation from the list box, and then select **Delete**.
- Modify an aggregation - select an aggregation from the list box, and then select **Modify**. The bottom panel populates with the aggregation values, and you can select or deselect the levels. When you finish modifying the aggregation levels, select **Apply** to save the changes. Select **Apply** to validate the aggregation. If the aggregation is not in hierarchy order, you will receive an error message.

Here are some guidelines for using Manual Tuning:

- Manual Tuning is available only for MOLAP cubes.
- Levels that are specified for an aggregation must follow at least one existing hierarchy.
- When you modify aggregations, you cannot modify the aggregation name.
- If the cube has an NWAY aggregation, it will display, but it cannot be modified or deleted.
- You cannot add duplicate aggregations in the Manual Tuning window. A duplicate aggregation has the same name as another aggregation or the same list of levels as another aggregation. When you add an aggregation to a cube and select **Apply**. Validation will occur to ensure that the aggregation to add is not a duplicate and that its levels follow a hierarchy order.

Note: If a cube has presummarized aggregations, the Modify function will not be active if a presummarized aggregation is selected. △

Using the Advanced Aggregation Tuning Plug-In

The Advanced Aggregation Tuning function provides automatic generation of cube aggregations. It is additive in nature. Aggregations created with the Advanced Aggregation Tuning function are added to the list of existing aggregations that may already be defined for the cube. It is available as a plug-in component for SAS OLAP Cube Studio.

Within the Advanced Aggregation Tuning plug-in, a generated list of aggregations is created. This list is then further processed to remove any aggregations that already exist in the cube. The Advanced Aggregation Tuning function includes two aggregation generation methods, Cross-Dimensional and ARM Analysis.

Cross-Dimensional up to number of levels:
: Cross-Dimensional tuning is recommended when a cube is first created and no ARM log exists yet. It constructs aggregations using the different combinations of levels across all hierarchies. You can use the spin selector to change the maximum number of levels to use in the hierarchies, starting at the top level of the hierarchy and going down in the hierarchy drillpath order.

Arm Analysis
: ARM Analysis is recommended for optimal cube tuning. It is the preferred method to generate aggregations for cube tuning. Arm Analysis constructs aggregations using the ARM log (created when the cube is queried with ARM logging turned on).

  ARM analysis requires that ARM logging be turned on and that an arm log already exist and be available in order for queries to be performed against the cube. The ARM log is used to deduce which aggregations to generate that will most likely have positive impact on query performance. It is only as effective as the amount of ARM data provided and whether the ARM log data truly reflects the cube query patterns going to be done. You can type in the name of log file or use the **`Browse`** button to select a file on a local machine. If you want to select a file that is on a remote machine, you can map the drive for remote access on your local machine. Generated aggregations will be listed in descending order by Count (the number of times that a query needed the aggregation). The **`Count`** and **`Elapsed Time`** (wall time) are displayed in the table with each generated aggregation.

With either aggregation generation method, you select the aggregations to add to the cube from a generated list.

Using PROC OLAP to Tune Aggregations

To modify an aggregation through PROC OLAP, use the DROP_AGGREGATION statement to delete the aggregation, and then use the AGGREGATION statement to define the new aggregation.

- DROP_AGGREGATION *level-name1* < *level-name2 ...level-nameN* > / NAME='*aggregation-name*' ;
- AGGREGATION *level-name1* / < NAME='*aggregation-name*' > ;

For more information about the DROP_AGGREGATION and AGGREGATION statements, see "The OLAP Procedure" on page 78.

Monitoring OLAP Server Performance

SAS OLAP Server performance is monitored and logged with the Application Response Measurement (ARM) interface. The ARM interface provides built-in logging capabilities and generates log records that indicate query content and query start and completion times. From this data, information regarding aggregation usage, individual query response times, or throughput can be determined. Traditionally, ARM enables system administrators to monitor application executions, run times, performance, and completion. SAS OLAP Server uses ARM to monitor

- application behavior
- user behavior and usage
- server loads
- cube optimization (query response time)
- cube metrics—counts of connections and queries.

For more information, see "Monitoring Performance Using Application Response Measurement (ARM)" in *SAS Language Reference: Concepts*.

Specifying Tuning and Performance Options in Cube Aggregations

When you build cubes, you can set various options that improve and optimize cube creation and query performance. These options can be set for all aggregations in a cube or for a specific aggregation. Additionally, these options can be set by using the PROC OLAP options or in SAS OLAP Cube Studio. These options are stored with the cube metadata in the SAS Metadata Repository.

Setting Options on the Cube Designer Wizard

In the Cube Designer - Generated Aggregations window, an Advanced button is provided for access to tuning options. Select the **`Advanced`** button to open the Performance Options window. There are two tabs for setting tuning options, the Global tab and Aggregation tab.

Global Tab

The global performance options are applied to all aggregations for the cube. These performance options include the

- amount of memory (in megabytes) that is available for aggregation creation
- maximum number of threads that are used to create an aggregation index
- number of aggregations to create in parallel
- partition size (in megabytes) of aggregation table partitions
- number of observations (in kilobytes) to include in the index component file segment
- location of index component files
- location of partitions in which to place aggregation table data
- aggregation tables that are stored in compressed format.

For specific information about these functions, see the Performance Options - Global tab in *SAS OLAP Cube Studio Help*.

Aggregation Tab

The aggregation-specific performance options are applied to an individual aggregation for the cube and override the global option settings for that aggregation. You can define and modify performance options for an aggregation or delete options for an aggregation. The aggregation-specific performance options include the

- partition size (in megabytes) of aggregation table partitions
- number of observations (in kilobytes) to include in the index component file
- location of index component files
- location of partitions in which to place aggregation table data
- aggregation tables stored in compressed format
- aggregations created with indexes

For specific information about these functions, see the Performance Options - Define Aggregation Options dialog box in *SAS OLAP Cube Studio Help*.

Setting Options on the Manual Tuning Function or the Advanced Aggregation Tuning Plug-in

When you add aggregations to a cube, using either the Manual Tuning or the Advanced Aggregation Tuning function, it is sometimes necessary to change the original settings for certain performance options to more optimal values. It is also possible to change the values without having to rebuild the cube.

The Options window enables you to specify certain performance options on the PROC OLAP statement when adding or modifying aggregations for a cube. On this window you can view the current values and change the values when needed. If you change one of the performance option values, the option will be added onto the PROC OLAP statement when the tuning code is submitted. The procedure will then add the new values into the metadata. When you first open the window, the current set values of the options will be listed. The following performance tuning options are available:

Number of aggregations to create in parallel	the number of concurrent aggregations that will be built. This value must be greater than or equal to zero. This is the equivalent of the PROC OLAP CONCURRENT option.
Maximum number of threads to use	the number of threads to use when creating aggregations. This value must be greater than or equal to zero and less than 65536. This is the equivalent of the PROC OLAP MAXTHREADS option.
Memory to use creating aggregations	the amount of memory to use when creating aggregations. This value must be greater than or equal to 32 and less than 10239. This is the equivalent of the PROC OLAP INDEXSORTSIZE option.

Note: If no aggregation changes are made or if the only changes made are deletions of aggregations, changes made to the tuning options will not be saved. △

Setting Options with PROC OLAP

You can set options for all aggregations in a cube or for a specific aggregation. To set options for all aggregations, set the options in the PROC OLAP statement. To set options for a single aggregation, set the options in the PROC OLAP - Aggregation statement. The options include

INDEXSORTSIZE=*n*
specifies the amount of memory in megabytes that is available when aggregations are created. The default is the system's available memory.

MAXTHREADS=*n*
specifies the maximum number of threads that are used to asynchronously create the aggregation indexes.

CONCURRENT=*n*
specifies the maximum number of aggregations to create in parallel.

WORKPATH=*pathname*
specifies one or more locations for temporary work files.

DATAPATH=('*pathname1*' ...'*pathnameN*')
specifies the location of one or more partitions in which to place aggregation table data.

INDEXPATH=('*pathname1*' ...'*pathnameN*')
specifies the locations of the index component files that correspond to each aggregation table partition as specified by the DATAPATH= option.

COMPRESS | NOCOMPRESS
specifies whether or not to store the aggregation tables in a compressed format on disk.

INDEX | NOINDEX
specifies whether or not to create the aggregations with indexes.

PARTSIZE=*partition-size*
specifies the partition size in megabytes of the aggregation table partitions and their corresponding index components.

SEGSIZE=*rows-per-segment*
specifies the number of observations (table rows) in kilobytes to include in the index component file segment.

Note: INDEXSORTSIZE=, MAXTHREADS=, and CONCURRENT= are only available in the PROC OLAP statement. △

For more information about these options, see "PROC OLAP Statement" on page 79 and "AGGREGATION Statement" on page 99.

Multiple Language Support and Dimension Table Translations

OLAP cube data is often generated in one language but needed in other languages. For example, a company's OLAP cube data might be stored in English, but users who speak Spanish and Turkish need access to it. So, the member values as well as the captions that are assigned to dimensions, hierarchies, levels, etc., need to be translated. Multiple language support is available only for cubes that are loaded from star schemas.

It is used to read your alternate locale data sets and create locale-specific metadata for use at query time. Query results are returned in the language of the requested locale.

You can specify language support when building a cube either in the Cube Designer Wizard or with PROC OLAP code. There are 56 possible language locales, and English is the default language.

SAS OLAP Cube Studio and Dimension Table Translations

In the Cube Designer - General window, select the **Advanced** button. If you selected Star Schema as the input type in the Cube Designer - General window, you will see the Dimension Table Translations tab. From the Available Languages/Locales list box, select the needed languages for the translation tables. The first language in the Selected Languages/Locales list box is the default language.

PROC OLAP and the USER_DEFINED_TRANSLATIONS Statement

The USER_DEFINED_TRANSLATIONS statement is used in conjunction with the DIMENSION statement options DIMTABLEMEMPREF= and DIMTABLELIBREF=. For more information, see the "DIMENSION Statement" on page 86.

SAS Servers and Character Encoding

If your server metadata contains characters other than those typically found in English, then you must be careful to start your server with an encoding= or locale= system option that accommodates those characters. For example, a SAS server started with the default US English locale cannot read metadata that contains Japanese characters. SAS will fail to start and log a message indicating a transcoding failure.

In general, different SAS jobs or servers can run different encodings (such as ASCII/EBCDIC or various Asian DBCS encodings) as long as the encoding that is used by the particular job or server can represent all the characters of the data being processed. In the context of server start up, this requires that you review the characters used in the metadata describing your server (as indicated by the server= objectserverparm) to ensure that SAS runs under an encoding that supports those characters.

Adding SAS System Options to a Cube

When you build an OLAP cube, it is often necessary to include additional SAS code that must run prior to the creation of the cube. This includes the creation of user-written formats, PROC statements, and format search paths for the formats that are used on input tables. The Advanced Cube Options window that is accessed from the Cube Designer - General window provides the two entry tabs, Submit SAS Code and Format Search Path. Both tabs provide entry fields for SAS code. You can enter any text in the fields. There is no validation of the text that is entered. However, error messages are sent to the SAS log.

The text is saved to the cube metadata in the SAS Metadata Repository and is used every time the cube is created. You can edit or remove the text after it is initially entered. Highlight the text and make any needed changes, or use the Delete key to remove the text. Select **OK** to save your changes.

Submit SAS Code	You can use this field to enter a PROC statement or any SAS code that you want to submit before the cube is created. SAS code is submitted before any format search path.

Format Search Path
: You can use this field to enter names of catalogs or libraries for the format search path. The catalogs and libraries must be separated by a blank and will be searched in the order in which they are listed. You use the SAS system option FMTSEARCH= here.

  Note: For more information about SAS formats, see "Formats" in *SAS Language Reference: Dictionary*. △

Note: When you build a cube with SAS OLAP Cube Studio, the format search path is saved with the cube metadata in the SAS Metadata Repository and used every time the cube is recreated. However, if you submit PROC OLAP code through a SAS session, outside of SAS OLAP Cube Studio, the format search path is ignored. PROC OLAP will not read the information from the SAS Metadata Repository or write the information to the SAS Metadata Repository. △

Synchronizing a Cube

The SAS OLAP Cube Studio — Synchronize Levels function enables you to synchronize a cube when the input table for an existing cube has encountered a column name change. This function finds the name differences between the cube and its input table and changes the hierarchy level names to match the input table column names. You can access the Synchronize Levels function in SAS OLAP Cube Studio from the **Actions** menu or by right-clicking a cube. If you are not connected to a workspace server when you select a cube, you will be prompted to select a workspace server.

Sometimes it is necessary to change a variable name in the input table that a cube is based on. However, this can result in the level names for an existing cube to be out of date. For example, if an input table for a cube has a hierarchy of Year, Quarter, Month, Day, and the column name "Month" is changed to "Mois" (the French spelling of Month), then the cube hierarchy level name "Month" will be out of sync with the input table's column name "Mois." The Synchronize Levels function enables you to update the hierarchy with the new month name.

The Synchronize Levels function obtains the cube metadata from the Object Metadata Repository and compares the names between the input table and the cube hierarchy. If a discrepancy is found, a new cube file and definition are created with the new level name. The level name of the existing cube is then updated to reflect the new column name.

The Synchronize Levels function is available for a cube if the cube physically exists and you have writemetadata permissions for that cube. In addition, if you are in a change-managed environment, you must be in the project repository.

Note: If the default Level caption is used (the Level name), then the Level caption is also updated. You need to update any existing reports or MDX queries that use the old level name. You also need to update any permission conditions or calculated members that reference the old level name. You cannot update an unregistered cube. △

Exporting and Importing Cubes

When administering SAS OLAP cubes it is sometimes necessary to copy or move the cubes. Some possible activities include the following:

- copying cubes from one environment to another
- archiving cubes for backup purposes
- moving cubes from a development system into production

SAS OLAP Server cubes are collections of files that are typically large and have a physical component (the files that make up the cube), and a metadata component (the cube's registration in a metadata repository). Both the cube files and registration need to be kept in sync when you copy or move cubes. The SAS OLAP Cube Studio Export Cube and Import Cube functions enable you to copy cube metadata from a source repository to a target repository, and if needed, to another server.

The Export Cube function extracts a cube's metadata from the source repository and saves it in a file that is specified by the user. All information about a cube, including its dimensions, hierarchies, levels, measures, notes, properties, calculated measures, aggregations, and security settings, will be extracted. The Import Cube function then enables you to save the cube metadata to another metadata repository on another metadata server.

Exporting Cubes

The Export Cube function extracts a cube's metadata from the current repository and saves it in a file that is specified by the user. All information about a cube, including its dimensions, hierarchies, levels, measures, notes, properties, calculated measures, aggregations, and security settings, will be extracted. You can launch the Export Cube function by right-clicking a cube object in the Cube Studio Navigation Tree and selecting **`Export Cube`**. This opens the Export Cube to File window.

When exporting cube metadata, you must provide a name for the file that you are exporting the metadata to. If you have not previously created an export file, then the default filename displayed will be user-default-directory\cubename.xml.

If you have previously created an export file, then the directory of the filename displayed will be the last directory that you exported cube metadata to. The filename will be composed of last-export-directory-used\cubename.xml. The name of the cube that you selected will be entered along with a .xml extension.

Note: The exported file should not be edited or changed by the user. △

Importing Cubes

The Import Cube function imports a cube's metadata from a previously exported file and enables you to save the cube metadata to another metadata repository on another metadata server. All information about a cube, including its dimensions, hierarchies, levels, measures, notes, properties, calculated measures, aggregations, and security settings, will be imported.

You can launch the Import Cube function by clicking **`Import Cube`** on the SAS OLAP Cube Studio Shortcut Bar. You can also select it from the SAS OLAP Cube Studio Tools menu. This opens the Import Cube from File window. In the **`Import file`** field, you can enter the name and location of the file that you will import. You can also click the **`Browse`** button to select a file. If you have not selected an import file before, then the selected file that is shown by default is user-current-directory\extractedcube.xml. If you have selected an import file before, then the last selected file is displayed by default.

In addition to these considerations, all connection points for the cube must exist in the target repository with the same names in order for the cube to import correctly. The following metadata connection points must exist:

- the OLAP schema for the cube
- all of the tables (input, dimension, drill-through, and aggregation) and columns that are associated with the cube. The table in the target repository must have the same Library name as the table that is located where the cube was exported from

- any UniqueKey, ForeignKey, and KeyAssociation objects required by star schemas
- the identities, groups, and permissions for the security settings

Note: If a cube with the same name as the import cube already exists in the OLAP schema, an error message appears. △

Managing Cube Data

After selecting a cube to import, you need to indicate whether you want to create new cube data, refer to the existing cube data, or manually move the cube. On the Cube Import Wizard — Manage Cube Data window, select one of the following options:

`Yes, the cube data will be created after import` You will create new cube data after the cube metadata is imported. This is the default setting.

`No, the cube data will not be created after import` New cube data will not be created. The cube will refer to the existing cube data from the cube that you selected or you will manually move the cube data to the new file locations.

Note: If the cube must be created after the import, the option **`No, the cube data will not be created after import`** will be disabled. This will happen if the imported cube metadata indicates that the cube has not been created. It will also be disabled if you have changed a path that is associated with an aggregation. This is because cube data cannot be moved if an aggregation path changes. △

Cube and Aggregation Path Settings

After selecting a cube to import, you can verify and edit path settings for both the cube and the cube's aggregations. On the Cube Import Wizard— Substitute Cube Paths window you can view and edit settings for both Global paths and Aggregation paths.

Global Paths The path values for the imported cube are displayed in the left column, under the heading Exported Value. You can choose to modify the Import Value.

Aggregation Paths The aggregation path values of the imported cube are displayed for each aggregation, including the default aggregation. You can choose to modify the Import Value.

When you have finished editing the cube data and path settings you can finalize the import process on the Cube Import Wizard — Wizard Finish window. In this window, you can review the filename and storage path of the file that you are importing. You can also review the cube name and the OLAP schema for the cube. Click **`Finish`** to import the file. The cube metadata will be imported into your current repository.

User Privilege Considerations

It is recommended that the person performing the cube export or import be the AdminUser or someone who has full access to the cube. This is because other users might have restrictions on parts of the cube that could result in a partial cube being exported and imported.

Repository Considerations

It is also recommended that all information about a cube be stored in the same repository and that you export and import a cube into the same type of repository. For example: If you export from a foundation repository, then you should import into a foundation repository. Or if you export from a custom repository, then you import into another custom repository. An export from one type of repository to a different type of repository (for example, foundation to custom) is not supported at this time. Also, export from and import into project repositories is not supported.

Creating Connection Points

Certain metadata must exist prior to importing a cube. All metadata connection points for the cube must exist in the target repository with the same names. If a metadata connection object is not found in the repository, then the import will fail. The following metadata connection points must exist:

- the OLAP schema for the cube - OLAP schemas can be created using the OLAP Schema wizard in SAS OLAP Cube Studio.
- the tables (input, dimension, drill through and aggregation) and columns that are associated with the cube - the table in the target repository must have the same Library name as the table that is located where the cube was exported from. Use the Source Designer in SAS OLAP Cube Studio, SAS Management Console, or SAS Data Integration Studio to load the table and column definitions.
- any UniqueKey, ForeignKey, and KeyAssociation objects required by star schema - keys can be created using the Property Sheet for a table in SAS Management Console or SAS Data Integration Studio.
- the identities, groups, and permissions for the security settings - security settings can be created using the Authorization Manager in SAS Management Console.

Note: Foreign key associations must exist for your fact tables. However, if the foreign key association does not exist on the fact table for your star schema cube, the import will succeed but the error will be detected when you build the cube or edit the cube metadata using the Cube Designer. △

File Naming Considerations

When exporting cube metadata, you must provide a name for the file that you are exporting the metadata to. If you have not previously created an export file, then the file will be saved in the user-default-directory. The filename will be composed of user-default-directory\cubename.xml.

If you have previously created an export file, then the file will be saved in the last directory that you exported cube metadata to. The filename will be composed of last-export-directory-used\cubename.xml.

When importing cube metadata, if you have not selected an import file before, then the selected file shown by default is user-default-directory\extractedcube.xml. If you have selected an import file before, then the last selected file is displayed by default.

Note: An existing OLAP cube cannot be renamed. △

Multi-Language Cubes

You can export and import multi-language cubes. However, only the dimension tables for the server language (the first language in the UDT statement) are verified for

registration. You must ensure that all the tables are present. If one of the tables is missing when the cube is imported, the import will still be successful, but the cube might not rebuild correctly.

Manually Copying and Moving SAS OLAP Cubes

After copying a cube's metadata using the Export and Import functions, you can choose to manually copy the cube files from one location to another instead of reusing the existing cube files or rebuilding the entire cube. Copying the physical components of a cube involves copying some or all of the cube files including MOLAP aggregation tables or ROLAP aggregation and drill-through tables. Specifically, you can manually copy the following:

Manually Copying or Moving SAS OLAP Cube Files

The core files of a SAS OLAP cube are created in a subdirectory that has the same name as the cube in the cube's path. The cube's path is specified with either of the following:

- the SAS OLAP Cube Studio, Cube Designer Wizard. In the Cube Designer - General window, enter a file path in the Path field.
- the PATH= option in PROC OLAP.

For example, if you create the cube "OrionStar" in the cube path myserver\testolap\testcubes, then the cube files are stored in the path myserver\testolap\testcubes\OrionStar. When you choose to copy or move the cube to your target cube path, you should perform the following steps:

1. Create a subdirectory with the cube's name. For example, if you want to copy or move the cube to otherserver\prodolap\prodcubes, you would create the directory otherserver\prodolap\prodcubes\OrionStar.
2. Copy or move the cube files to the target cube subdirectory. This includes the following files:
   - cubename.cube (in our example; OrionStar.cube)
   - captiontrees
   - fmtcaps
   - mbrtrees
   - propcaps (does not exist for all cubes)

You can use standard operating system functions to copy or move the files.

Note: You should ensure that the target files have the same operating system permissions as your original files. △

Manually Copying MOLAP Files

SAS 9.1.3 OLAP cubes store aggregated data values in tables that have the same file structure as SAS Scalable Performance Data (SPD) Engine tables. These files cannot be moved using operating system commands. They must be moved by the SAS PROC COPY utility. This is necessary in order to correctly update the internal structure of the SPD files as they store path information in their header portion.

By default, the cube's aggregation table and index files are stored in the cube subdirectory in the cube's path. However, you can specify alternate locations for the data and index portion of the aggregation tables. For more information, see "Changing a Cube's Data and Index Paths" on page 62.

To copy the aggregation files for the OrionStar cube (assuming the aggregation files are in the cube path), you assign libref IN to the original location and the libref OUT to the new location. Both LIBNAME statements should use the SPD Engine. You can then use the PROC COPY statement to copy the files:

```
libname in spde "\\myserver\testolap\testcubes\OrionStar";
libname out spde "\\otherserver\prodolap\prodcubes\OrionStar"
proc copy in=in out=out;
run;
```

If you have specified an alternate location for the data and index files as described in "Changing a Cube's Data and Index Paths" on page 62, then you must use the INDEXPATH= and DATAPATH= options on the target LIBNAME statement (that is, the OUT libref):

```
libname in spde "\\myserver\testolap\testcubes\OrionStar";
libname out spde "\\otherserver\prodolap\prodcubes\OrionStar"
indexpath=("\\otherserver\olapindexes") datapath=("\\otherserver\olapdata");
proc copy in=in out=out /* move */;
run;
```

There are some specific restrictions that apply to using the PROC COPY statement. You can only copy or move cube data by using PROC COPY under the following conditions:

- if you have not specified a specific index path or data path by clicking the Advanced button in the Generated Aggregations window. In this case the aggregations are stored in the cube path and will be moved with the PROC COPY statement.
- if you have specified a global index path or data path only in the Performance Options window. All aggregation files will be stored in the global path and will be moved with the PROC COPY statement so long as you have specified INDEXPATH= or DATAPATH= on the SPDE LIBNAME statement.

Note: If you have specified INDEXPATH= or DATAPATH= on specific aggregations, you cannot use PROC COPY to move your MOLAP aggregation tables. You will need to recreate the cube, using the short form of PROC OLAP, in order to get the aggregation files in the correct location. △

Manually Copying ROLAP Files

SAS OLAP cubes can use ROLAP tables for the following uses:

- drill-through tables
- aggregation tables
- input data (if you used the NO_NWAY option)
- format catalogs (used by any of the above)

These tables can be stored in many formats including SAS tables, SPD Engine tables, SPDS tables, and tables in external RDBM systems. You can use PROC COPY to copy or move SAS tables, SPD Engine tables, and SPDS tables. You can use standard operating system functions to copy or move tables in external RDBM systems.

When you copy or move any of these files, you should verify that your target SAS OLAP Server has access to those files. In addition, you will need to change the LIBNAME specifications to point to the new file locations. Libnames for an OLAP server are allocated by using an AUTOEXEC file during the SAS OLAP Server invocation or by using pre-assigned libraries.

Changing a Cube's Data and Index Paths

In the Cube Designer wizard, the Performance Options window (for Generated Aggregations) contains two settings that you can modify:

- **Location of index component files**
- **Location of partitions in which to place aggregation table data**

You can access the Performance Options window by clicking the **Advanced** button in the Cube Designer - Generated Aggregations window. Select the **Global** tab. These options are comparable to the INDEXPATH= and DATAPATH= options in PROC OLAP. For more information about aggregation performance options, see "Performance Options - Global Tab" in the *SAS OLAP Cube Studio Help*.

Accessing OLAP Cubes from SAS: SQL Pass-Through Facility for OLAP

The SQL Pass-Through facility enables a SAS user to connect to an OLAP server and execute cube queries within the PROC SQL environment. PROC SQL establishes a connection to an OLAP server by using the PROC SQL CONNECT statement.

After a connection is made to the OLAP server, multiple queries can be submitted by using the OLAP query language, Multidimensional Expressions (MDX). These queries are run against existing OLAP cubes. A PROC SQL query is then closed after all observations (rows) of data are returned.

To disconnect from the server, you must submit the PROC SQL DISCONNECT statement.

The main function of the SQL Pass-Through facility for OLAP is to query data, but MDX commands can be submitted that create named sets, globally scoped sets, drill-through paths, and other OLAP components. Additionally, named sets that are created by using MDX can then be used to run queries. It is important to note the following conditions:

- These sets are only available during the current PROC SQL session.
- Other PROC SQL sessions cannot access or reference these sets.
- After a PROC SQL connection is disconnected, any session-created sets are discarded.

Any libraries that are required for cube queries are obtained from the metadata repository when the cube is first loaded on the server. These libraries are dynamically assigned when PROC SQL establishes a connection to an OLAP server.

Conversion Issues

OLAP cube data is multidimensional and flexible in regard to data name lengths and restrictions. However, when PROC SQL sends a query to the OLAP server, data is returned in a flattened, tabular format that contains rows (observations) and columns (variables).

The SAS OLAP Server has unique naming conventions that specify valid column names, lengths, and types. Column names that are returned from SAS OLAP can contain characters (periods, spaces, brackets) and can be unrestrained in length. Additionally, OLAP types can be variable-length strings, floating-point numbers, or integers. This differs from SAS data set naming conventions, and some conversion is necessary.

VALIDVARNAME

The SQL Pass-Through facility supports the existing SAS option VALIDVARNAME. You can specify the VALIDVARNAME option to control variable names. The current default setting for VALIDVARNAME is V7, and variable names can be a maximum of 32 characters in length. Each variable must start with a letter or the underscore character and can contain letters, underscores, and numbers. Uppercase and lowercase letters are also allowed.

When converting column names to SAS variable names, the SQL Pass-Through facility for OLAP will

- truncate the column name to the maximum size that is allowed.
- replace any invalid characters with an underscore.
- use a numeric suffix to differentiate between duplicate variable names that are generated during the data conversion.

Note: For additional information about naming restrictions for SAS OLAP Server, see "Naming Guidelines for SAS OLAP Server" on page 108. △

Note: For further information about the VALIDVARNAME= system option, see "VALIDVARNAME=System Option" and "Names in the SAS Language" in the *SAS Language Reference: Dictionary*. △

Data Types

OLAP query results that contain member names or strings are converted to a fixed length CHAR type. All OLAP numeric types are converted to standard SAS numeric types (8-byte floating point). Missing values are handled by standard SAS conventions.

PROC SQL Syntax

Here is an example of the basic syntax that is used to connect to an OLAP server and execute a cube query:

```
proc sql;
 connect to dbms-name (connection options);
 execute (MDX query);
 select . . . from connection to remote | alias (dbms-query);
 disconnect from dbms-name;
 quit;
```

CONNECT TO REMOTE
establishes a connection to a remote DBMS or to remote SAS data through a SAS server. This statement is required. Remote SQL Pass-Through (RSPT) does not support implicit connection.

DBMS=dbms-name
specifies the name of the remote DBMS that you want to connect to. For SAS OLAP Server, the dbms-name is *saseolap* or *olap*.

DISCONNECT FROM REMOTE | alias
ends the connection to the remote DBMS or to the SAS SQL processor in the server SAS session.

EXECUTE (SQL-statement) BY REMOTE | alias
specifies an SQL statement to be executed by the SAS SQL processor or by the remote DBMS in the server SAS session.

SELECT . . . FROM CONNECTION TO REMOTE | alias (dbms-query);
specifies the connection to the remote SAS SQL processor or the remote DBMS as the source of data for the SELECT statement and the recipient of the dbms-query.

Here are the bridge server *connection-options*:

HOST=*machine-name*
specifies either the DNS name or the IP address of the machine that is hosting the OLAP server.

PORT=*port-number* | SERVICE=*service-name*
either the port-number or service-name is required. The *port-number* specifies the numeric value of the port on which the OLAP server resides. The *service-name* is used to look up the port number of the machine that is hosting the OLAP server.

USER=*userid*
a string that specifies the user's identification for the specified OLAP server. If included, this option is enclosed within parentheses with the required arguments and any other options.

PASS=*password*
a string that specifies the password for the user who is identified with the USER= option. If included, this option is enclosed within parentheses with the required arguments and any other options.

This is the COM server *connection-option*:

machine-name
specifies either the DNS name or the IP address of the machine that is hosting the OLAP server. This is the only argument that is used for COM server connections.

Note: For detailed information about PROC SQL syntax see "Overview of the Pass-Through Facility" in *SAS/ACCESS for Relational Databases: Reference* and "Syntax for Remote SQL Pass-Through (RSPT) Facility" in *SAS/SHARE User's Guide*. △

SQL Pass-Through Example

In the following example, PROC SQL connects to the pass-through facility for OLAP to create a new data set named temp, which contains all the variables that are returned from the multidimensional expression (MDX) defined in the SELECT query. The OLAP server returns query results in a tabular format known as a flattened rowset. The table rows become the observations of the output data set, and the table columns become the variables. After all the rows are returned, PROC SQL closes the query. The server connection is terminated when the program encounters a DISCONNECT statement or when the PROC SQL step ends.

Note: Because the OLAP server does not impose the same restrictions on column names, types, and lengths that SAS imposes on data sets, some conversion might be required. △

```
100        proc sql;
101           connect to olap (host=localhost service=olap1);
102           create table temp as select * from connection to olap
103           (
104              select { dealers.dealer.members } on 0,
105                     { [cars].[Car].members,
106                       [cars].[Color].members } on 1
107              from mddbcars
```

```
108          );
109         disconnect from olap;
110         quit;
```

Specifying GIS Map Information for a Dimension

The SAS OLAP Cube Studio Specify Map function allows you to store ESRI Geographic Information System (GIS) spatial map information in the SAS Metadata Repository. This GIS information can then be read by the SAS OLAP Server and returned during a cube query. You can set up the ESRI ArcGIS server information in the metadata repository by using the Map Service Manager plug-in in SAS Management Console. Users can access the functionality through the SAS Web OLAP Viewer and SAS Web OLAP Viewer for Java.

The Specify Map function enables you to identify a geography-based dimension and then assign ESRI spatial map information to that dimension. To define GIS information for a SAS OLAP cube, you identify a dimension as a geographic-type dimension (GEO) in the Dimension Designer - General window. You can have only one GEO dimension per cube. After a dimension has been marked as a GEO-type dimension, the Specify Map button becomes active on the Cube Designer - Dimensions window.

Note: The Specify Map button is dimmed until a GEO dimension has been specified for a cube. △

The Specify Map window enables you to assign ESRI spatial map information to levels of the GEO-type dimension. You can add map information to an existing cube, modify map information for a cube, or delete the map information from a cube. When ESRI map information is added to a cube, the property objects for the mapped cube levels are listed in the Cube Designer's Member Property window. From here you can modify the format, caption, and description for the member property. These member properties are named SAS_SPATIAL_ID by default.

Note: If a dimension is changed from type GEO to Standard or Time, all the map information is removed. △

For further information on the Specify Map window, see the SAS OLAP Cube Studio Help.

Specifying Calculated Members

After you have built a cube, you might find it necessary to add members or measures to the cube. You can add calculated members to a cube in the following ways:

- manually by submitting PROC OLAP code with the DEFINE statement. See "DEFINE Statement" on page 102.
- using the Calculated Members plug-in in SAS OLAP Cube Studio.

The Calculated Members plug-in enables you to add new members for the Measures dimension. A calculated member is a definition (for a dimension member) that you create and store with the cube. The calculated member value is generated later during query time. You can also define a calculated member as a measure.

Using the Calculated Members Plug-in

The Calculated Members plug-in can be accessed from the SAS OLAP Cube Studio Shortcuts Bar or from the **Tools** menu. Click **Calculated Members** to open the Cube List dialog box. In this dialog box, select a cube from the repository tree and select **OK**. The Calculations for cube dialog box opens.

On the Calculations for cube dialog box you can add new calculated members or modify existing members for the selected cube. The list box displays all calculated members that are defined in the metadata for the selected cube. Select **Add** to launch the New Member Wizard and define a new calculated member.

To modify an existing member, select the member and then click **Edit**. The Edit a Calculated Member dialog box opens.

Selecting Calculation Types

When you click **Add** in the Calculations for cube dialog box, the New Member Wizard is launched. On this window, you select the type of calculation that you want to create. You can select one of the following types:

- Simple Calculations to create a simple calculation formula. See "Simple Calculations" on page 66.
- Time Analysis Calculations to create a time-based calculation formula. See "Time Analysis Calculations" on page 67.
- Custom Calculations to enter a custom calculation formula. See "Custom Calculations" on page 67.

Simple Calculations

On this window, you select the variables to create a simple calculation. You can select one of the following calculation types:

- Sum
- Difference
- Ratio
- Percent Increase
- Percent Decrease
- Distinct Count

The **Formula** drop-down lists for the members are populated with all measures, including calculated measures. The **Formula** panel changes depending on the calculation-type radio button you select. When a formula is selected, the **Formula** panel is populated with drop-down lists appropriate for the selected calculation. A member must be selected in each list. The member selections for the different formulas are preserved across formulas within a type.

If you select **Distinct Count**, a selection tree is displayed. Each dimension has a node that expands into its hierarchies and then into the levels for each hierarchy. One variable in the hierarchy tree must be selected.

For further information on the Simple Calculations window, see the SAS OLAP Cube Studio Help.

Time Analysis Calculations

On this window, you select the variables to create a Time calculation. You can select one of the following calculation types:

- Opening Balance
- Closing Balance
- Rolling Total
- Average Over Time
- Compare Parallel Periods
- Compare Consecutive Periods

The **Formula** panel changes depending on the time-calculation radio button that you select. The **Existing measure** drop-down list is populated with all measures, including calculated measures. The **Time period** drop-down list is populated with members from the Time dimension. A member must be selected in each list. The member selections for the different formulas are preserved across formulas within a type.

Note: This option is not available if there are no Time-type dimensions for the cube. △

For further information on the Time Analysis Calculations window, see the SAS OLAP Cube Studio Help.

Custom Calculations

On this window, you enter the variables to create a custom calculation. You can enter the following:

Parent dimension	specifies a parent dimension for the new calculated member that you are creating. The default dimension is Measures. If you select a dimension other than Measures, this field will initially be populated with the unique name for the default hierarchy's all member.
Parent member	specifies a parent member for the new calculated member. Use the **Browse** button to display a tree of valid members for the dimension. If you select the **Browse** button and the cube physically exists, you will be prompted to log on to an OLAP server, if you haven't already. *Note:* If the cube exists, you must connect to the OLAP server so that the valid members can be retrieved from the server. If the cube does not exist or a connection cannot be made to the OLAP server, the valid members displayed in the tree will be the all members for each of the hierarchies for the dimension. △
Name	specifies a name for the custom calculation.
Formula	specifies an MDX formula for the new calculated member.

Click **Verify** to validate the MDX formula entered in the **Formula** text box. This function is only available for cubes that physically exist. When you select **Verify**, the Log on to an OLAP server dialog box opens. Select a server and enter a user ID and password. A connection will be made to the selected OLAP server. Click **Clear** to clear the contents of the **Formula** text box.

The **Generated MDX** text area allows you to view the MDX code that currently exists for the selected cube and any new MDX code that is entered in the **Formula** text box.

For further information on the Custom Calculations window, see the SAS OLAP Cube Studio Help.

CHAPTER

5

Using SAS OLAP Cubes

Using a Cube with ADO MD

Applications gain access to SAS OLAP cubes through ADO MD. *ADO MD* is an industry standard programming interface to multidimensional data. It offers the same functionality as OLE DB for OLAP but in a simpler programming model. Accessing SAS OLAP cubes through ADO MD requires the SAS OLAP Data Provider, which is a component of SAS Integration Technologies. The SAS OLAP Data Provider is installed with the SAS Integration Technologies Client for Windows 9.1. See the *SAS Data Providers: ADO/OLE DB Cookbook* for more information about IOM data provider usage with ADO MD.

Using a Cube with OLE DB for OLAP

In addition to ADO MD, applications gain access to SAS OLAP cubes through *OLE DB for OLAP*, an industry standard set of programmable Component Object Model (COM) interfaces that expose multidimensional data. For SAS OLAP cubes, the OLE DB interfaces are exposed by the SAS OLAP Data Provider, a component of SAS Integration Technologies. The SAS OLAP Data Provider enables applications to perform data analysis by providing a means to view schema information, submit MDX queries, and retrieve results. The SAS OLAP Data Provider is installed with the SAS

Integration Technologies Client for Windows 9.1. See the *SAS Data Providers: ADO/OLE DB Cookbook* for more information about IOM data provider usage.

Using a Cube with Additional SAS Products

SAS Products That Use SAS OLAP Cubes

The following additional SAS products access SAS OLAP cubes:

- "SAS Enterprise Guide" on page 70
- "SAS AppDev Studio" on page 70
- "SAS Information Map Studio" on page 71
- "SAS Web Report Studio" on page 71
- "SAS Web OLAP Viewer" on page 72
- SQL Pass-Through Facility for OLAP. For more information, see "Accessing OLAP Cubes from SAS: SQL Pass-Through Facility for OLAP" on page 62.

  Note: The SQL Pass-Through Facility for OLAP does not require additional licensing. △

See the product help and documentation for these products for information on how to access SAS OLAP cubes.

SAS Enterprise Guide

SAS Enterprise Guide is a project-oriented Windows application that is designed to enable quick access to much of the analytic power of SAS software for statisticians, business analysts, and SAS programmers.

SAS Enterprise Guide is an ODBO-compliant OLAP Viewer. When accessing OLAP cubes via ODBO, SAS OLAP Server acts as an Open OLAP Provider and Enterprise Guide is an Open OLAP Consumer.

SAS Enterprise Guide provides an OLAP-specific tool called the *OLAP Analyzer*. The OLAP Analyzer (formerly named the MDDB Viewer) has been simplified by the development of a new user interface and wizards for common tasks. You can view the detailed data that makes up any tuple (that is, a pairing of items from two dimensions) in your OLAP cube. In addition, you can add calculated members, or formulas, to a dimension in an OLAP cube. The calculated members can be computed from other members or values and must return either strings or numeric values.

SAS Enterprise Guide also enables you to create SAS Stored Processes and to store that code in a repository that is available to a SAS Stored Process Server. (Stored processes are SAS programs that are stored on a server and are executed by client applications.) Stored processes are used for Web reporting and analytics, among other things.

For more information about SAS Enterprise Guide, see the SAS Enterprise Guide Help, which is available from within the product.

SAS AppDev Studio

SAS AppDev Studio provides a single interface for the development of thin- and power-client business intelligence applications. SAS AppDev Studio supports every

major Web standard on both the server and the client side. CIOs can choose SAS as their standard for business intelligence with the knowledge that the analytical power of SAS can be deployed throughout the organization, regardless of the applications development and information distribution architecture.

Using tools that are part of SAS AppDev Studio, you can develop applet-based or servlet-based (including JavaServer Pages) OLAP applications.

For information, examples, and reference information on this product, consult the following sources:

- SAS AppDev Studio Help
- SAS AppDev Studio Developer's Site at the SAS Customer Support Center, **`support.sas.com/rnd/appdev`**.

SAS Information Map Studio

SAS information maps can translate business questions into the necessary MDX code to access SAS 9.1 OLAP structures. SAS Information Map Studio is a Java application that enables data modelers and data architects to create and manage SAS Information Maps, which are business metadata about your physical data. Information maps enable you to surface your data warehouse data in business terms that typical business users understand, while storing key information that is needed to build appropriate queries. Information maps provide the following benefits:

- Users are shielded from the complexities of the physical data.
- Data storage is transparent to the consumers of information maps, regardless of whether the underlying data is relational or multidimensional, or whether the data is in a SAS data set or in a third-party database system.
- Business formulas and calculations are predefined, which makes them usable on a consistent basis.
- Users can easily query data for answers to business questions without having to know query languages.

For more information about SAS Information Map Studio, see the SAS Information Map Studio Help, which is available from within the SAS Information Map Studio product.

SAS Web Report Studio

SAS Web Report Studio is a Web-based application that enables you to create, view, and organize reports. You can use SAS Web Report Studio to perform these OLAP-specific tasks:

- drill and expand tables and graphs
- support ragged and unbalanced hierarchies
- pivot individual crosstab dimensions
- switch dimensions and measures with Data Selection box
- synchronize report components to display a common drill state or have them remain independent

In addition to OLAP-specific tasks, you can perform the following tasks:

Creating reports
: Beginning with a simple and intuitive view of your data provided by SAS Information Maps (created in SAS Information Map Studio), you can create reports based on either relational or multidimensional data sources. You can use the Report Wizard to

	quickly create simple reports or the Edit Report view to create sophisticated reports that have multiple data sources, each of which can be filtered. These reports can include various combinations of list tables, crosstabulation tables, and graphs. Using the Edit Report view, you can adjust the style to globally change colors and fonts. You can also insert stored processes that take the results from a block of SAS code and embed those results directly into a report.
Viewing and working with reports	While viewing reports using a thin client (a Web browser), you can filter, sort, and rank the data that is shown in tables, crosstabulations, and graphs. With multidimensional data, you can drill down on data in crosstabulations and graphs and drill through to the underlying data.
Organizing reports	You can create folders and subfolders for organizing your reports. Information consumers can use keywords to find the reports that they need. Reports can be shared with others or kept private.
Printing and exporting reports	You can preview a report in PDF and print the report, or save and e-mail it later. You have control over many printing options, including page orientation, page range, and size of the tables and graphs. You can also export data as a spreadsheet and export graphs as images.

For more information about using SAS Web Report Studio, see the SAS Web Report Studio Help, which is available from within the product. For information about administrative tasks associated with SAS Web Report Studio, see the *SAS Intelligence Platform: Web Application Administration Guide*, which is available at **`support.sas.com`**.

SAS Web OLAP Viewer

SAS Web OLAP Viewer is a data exploration tool designed for business analysts who need to look at large volumes of data quickly from varying perspectives. It lets business users look at data from multiple angles, drill into detail data, and save their information for easy Web-based report creation.

SAS Web OLAP Viewer surfaces SAS 9 OLAP cubes, OLAP Information Maps, and OLAP-based data explorations. It provides an easy-to-use interface from which you can select a data source, view the data, and customize your view with features such as sorting and filtering. With SAS Web OLAP Viewer you can perform these tasks:

- apply filters and rankings to cubes
- calculate new measures
- sort data values
- add totals and subtotals
- save and reload changes to your data view
- create PDF for printing
- export to Excel

There are two versions of the SAS Web OLAP Viewer: SAS Web OLAP Viewer for Java and SAS Web OLAP Viewer for .NET. For further information on SAS Web OLAP Viewer, see the SAS Web OLAP Viewer Help.

SAS Web OLAP Viewer for Java

SAS Web OLAP Viewer for Java is a standalone, browser-based, data exploration tool. It is a Java Web application and is available as part of the SAS BI Server or SAS

Enterprise BI Server. It provides you advanced OLAP navigation and the ability to use ESRI maps to visualize data. You can access OLAP data with SAS Web OLAP Viewer for Java in two ways:

- By using SAS Information Maps, you can control the way users access data sources.
- By using direct access to cubes, you can reduce the amount of preparation work that is necessary before data access.

SAS Web OLAP Viewer for Java also enables you to design new calculated measures at runtime. You can then publish your views as reports in SAS Web Report Studio.

SAS Web OLAP Viewer for .NET

SAS Web OLAP Viewer for .NET is a Microsoft .NET Web application. It provides you advanced OLAP navigation and the ability to edit MDX queries directly. SAS Web OLAP Viewer for .NET can access any standards-compliant OLE DB for OLAP data source, such as the following:

- SAS OLAP Server
- Microsoft Analysis Services

SAS Web OLAP Viewer for .NET also enables you to design new calculated measures at runtime. Calculated measures let you use the full power of MDX to create calculations based on measures within your OLAP data source. You can distribute your results in the following ways:

- exporting slices from your OLAP data source to flat tables or Microsoft Excel spreadsheets
- exporting slices to SAS Enterprise Guide for further analysis using analytics or forecasts

Using a Cube with Third-Party Clients

The SAS OLAP Server exposes multidimensional data through OLE DB for OLAP interfaces. Supporting these industry standard interfaces enables the SAS OLAP Server to integrate with third-party clients. The user interfaces for these clients vary widely. To ensure successful integration with the SAS OLAP Server, usage guidelines for some third-party clients have been established.

Microsoft Excel 2000 and Excel 2002 PivotTable

To view SAS OLAP cubes in Microsoft Excel, you must identify the server where your cubes are stored and the cube you want to analyze with the *PivotTable and PivotChart* Wizard.

1. In Microsoft Excel, select **Data ▶ PivotTable and PivotChart Report**. This opens the PivotTable and PivotChart Wizard.
2. At the PivotTable and PivotChart Wizard, select the radio buttons for **`External data source`** and **`PivotTable`**. Select **`Next`**.
3. In the PivotTable and PivotChart Wizard window, select **`Get Data`**. This opens the Choose Data Source window.
   - On the OLAP Cubes tab, select **`<New Data Source>`**.
   - Select **`OK`**. This opens the Create New Data Source window.
     - In field 1, enter the name that you want to associate with the cube data you are accessing.

Note: This is the name that Excel uses to store your work. △

- In field 2, select **SAS OLAP Data Provider 9.1**.
- In field 3, select the **Connect** button. This opens the Data Link Properties window. In the Data Source field, enter the name of the SAS 9 OLAP Server you are accessing. If necessary, enter your user ID and password for accessing the server. Select the **SAS Protocol** (Bridge, Com, or Corba). If you select **Bridge**, then you must specify the SAS Service Name/Port. Select **OK**.
- If the connection to the server is successful, field 4 will be active.

  Note: A successful connection depends on several factors such as accurate data source and port information, accurate user account information, and whether the server is running and can be accessed. For details about SAS Protocol, SAS Service Name, and other connection properties, see "SAS OLAP Provider Connection Properties" in the *SAS Data Providers: ADO/OLE DB Cookbook*. △

  Select the cube that you want to analyze in Excel.

  Note: If field 4 is active, but there are no cubes shown in the drop down box, this means the OLAP server was unable to locate any cubes. A possible cause is that your OLAP server is not associated with the correct OLAP schema. You can determine the OLAP schema assigned to your OLAP server from the SAS Management Console. Then in SAS OLAP Cube Studio, verify that the OLAP schema has cubes. For more information about OLAP schemas see the *SAS Intelligence Platform: Data Administration Guide*. △
- You must select the radio button **Save my user ID and password in the data source definition** when you connect to a secure server. If you select the radio button, then a message window opens that informs you how the user ID and password will be stored in the data source definition. At this point you must confirm your selection of the radio button. There are variations in performance between Microsoft Excel 2000 and Microsoft Excel 2002. If you do not select the radio button, then when you connect to a secure server, the connection will fail in one of these ways:
  - In Microsoft Excel 2000, you will return to the Choose Data Source window in the PivotTable and PivotChart Wizard.
  - In Microsoft Excel 2002, the Data Link Properties window reloads, which allows you to re-enter the data source information.
- When you finish entering information in the Create New Data Source window, select **OK**. At this point Excel saves the data as a Microsoft query (OQY) file that you can later reference and load in Excel.

After completing the fields in the Create New Data Source window, you return to the Choose Data Source window. You see your new data source listed on the OLAP Cubes panel. Select **OK**. This returns you to the PivotTable and PivotChart Wizard window. Select **Next**. This loads the PivotTable and PivotChart Wizard.

4 At On the PivotTable and PivotChart Wizard, select where you want to put the PivotTable. You can choose to place the PivotTable in a New worksheet or in the Existing worksheet. Select **Finish**. The PivotTable is loaded.

5 At the worksheet and PivotTable menu, select cube data items to populate the columns and rows of the PivotTable. Use the drag-and-drop selection method to move the data buttons to the PivotTable. For the PivotTable data area, select the data buttons that are measures of the cube data.

6 From this point, use standard Microsoft Excel functionality to view cube data.

Note: You must have Microsoft Query installed to view OLAP cubes in Microsoft Excel. △

Saving a PivotTable as a Web Page

When you save a PivotTable as a static Web page in Microsoft Excel 2002, the user password is not automatically saved with the HTML file that is generated. For Excel 2002 PivotTables that are connected to a secure OLAP server, this can result in an error message when the HTML file is loaded into Internet Explorer.

In Microsoft Excel 2002, when you select**File ▶ Save as Web Page** a message window displays stating that the user password will not be saved. You then have an opportunity to continue saving the HTML file or not. If you choose to save the HTML file without the password, an error message might be displayed when the user tries to load the HTML file in Microsoft Internet Explorer. To prevent this error message, you can manually edit the HTML file to include the user password. For example, in this connection tag, you would add the password option, **`password=mypass;`** to the <Connection> string.

```
<Connection>Provider=sas.OLAPProvider.9.1;User ID=&userid;Data
Source=SAS OLAP Provider Server;Location=&location;
Mode=ReadWrite|Share Deny None;SAS Logical Name="";
SAS Machine DNS Name=&SASMachineDNSName ;
SAS Port=&Port;SAS Protocol=2;
SAS Server Type=2</Connection>
```

Microsoft Office Web Components 2000 and 2002 PivotTable

For the Microsoft Office Web Components 2000 and 2002 PivotTable user interface, the PivotTable connection string must specify the provider and data source. Specifically, the connection string must specify

- Provider=sas.OLAPProvider.9.1
- the data source as a URL/URI style string that contains all the necessary connection properties:

  ```
  olapServerName&Property1=value&property2=Value&property3=value
  ```

  The string must begin with the name of the OLAP Server to which you are connecting. To list additional connection properties, follow the server name with "&" and list "property name=property value" pairs, delimited by "&".

  Note: All connection properties, with the exception of provider, must be specified within the data source string. △

  Note: For a description of these properties see the *SAS Data Providers: ADO/OLE DB Cookbook*. In particular, see the sections "Connections and Data Sources" and "Connecting to a Remote SAS OLAP server." △

Here is an example of an HTML page using a Microsoft Office 2000 PivotTable in conjunction with the SAS OLAP Server. The data source string specifies an OLAP server with the name mktg.unx.com, a SAS port of 6176, and the ProtocolBridge for SAS protocol defined as 2.

```
<HTML>
<BODY ONLOAD = "Setup_PT()">
<OBJECT ID="PivotTable1"
             CLASSID="CLSID:0002E520-0000-0000-C000-000000000046"
```

```
            style="HEIGHT: 500px; WIDTH: 500px">
</OBJECT>
<SCRIPT language=VBScript>
   Sub Setup_PT()

constr="Provider=sas.OLAPProvider.9.1;Data
   Source=mktg.unx.com&SAS Port=6176
&Location=localhost&SAS Protocol=2''
      PivotTable1.ConnectionString=constr
      PivotTable1.DataMember="CAMPAIGN"
      PivotTable1.ActiveView.AutoLayout

   End Sub
</SCRIPT>

</BODY>
</HTML>
```

For Microsoft Office 2002 PivotTable Web Components, use CLASSID="CLSID:0002E552-0000-0000-C000-000000000046".

ProClarity Professional

You can access SAS OLAP cubes from within ProClarity Professional. The following steps show you how to load a SAS OLAP cube for analysis:

1. From the File menu, select **Options ▶ OLAP Provider ▶ Change ProviderSAS OLAP Data Provider 9.1 ▶ OK**On the OLAP Provider tab of the Options window, SAS OLAP Data Provider 9.1 should now appear as the current provider. Select **OK**.
2. From the File menu, select **Open Cube**. This opens the Connect window.
3. In the **Server** field, enter the name of the SAS OLAP server that you are connecting to as well as the user name and password. Select **OK**. This loads the Open Cube window.
4. In the Open Cube window, select a cube to open. Select **OK**. The cube is loaded into the ProClarity workspace.
5. From this point, use standard ProClarity functionality to manipulate the SAS OLAP cube.

Note: The ProClarity interface currently enables you to specify the data source that you are connecting to as well as the user name and password. However, it does not allow you to specify other provider properties that might be necessary to establish a server connection such as SAS Port or SAS Protocol. For further information about how to set additional SAS Provider properties, see the *SAS Data Providers: ADO/OLE DB Cookbook*. △

Note: In ProClarity you can access and view measures by selecting **View ▶ MDX Editor**. The cube measures are displayed in the lower metadata window. However, when viewing SAS OLAP cubes in ProClarity, user-defined measures are not recognized. △

APPENDIX

1

The OLAP Procedure

The OLAP Procedure

The OLAP procedure is one of the tools in SAS 9.1 that you can use to create and delete cubes, and add and delete aggregations.

Note: You can also use the Cube Designer wizard to maintain OLAP cubes. The Cube Designer wizard can be launched from SAS Data Integration Studio and SAS OLAP Cube Studio. Help on using the wizard to build cubes is available from within both applications. △

In addition to the basic cube creation tasks, PROC OLAP also enables you to

- build cubes with ragged hierarchies
- control options that can be used to optimize cube creation and query performance
- specify data set options on detail, fact, dimension, and drill-through tables
- create TIME dimensions
- design dimensions that have more than one hierarchy
- define global calculated members and named sets
- include SAS code when you submit PROC OLAP in batch mode
- read alternate locale data sets and create locale-specific metadata for use at query time

Syntax: OLAP Procedure

PROC OLAP < *option(s)*>;

METASVR OLAP_SCHEMA='*schema-name*' < *option(s)*>;

DIMENSION *dim-name* HIERARCHIES=(*hier–name*) <*option(s)*>;

HIERARCHY *hier-name* LEVELS=(*level–name1*<*level–name2 ... level-nameN*>) <*option(s)*>;

LEVEL *level-name* <*option(s)*>;

PROPERTY *prop-name* LEVEL=*level-name*<*option(s)*>;

MEASURE *measure-name* STAT=*statname* < *option(s)*>;

AGGREGATION *level-list* < /*option(s)*>;

DROP_AGGREGATION *level-name1* < *level-name2 ...level-nameN*> / NAME='*aggregation-name*' ;

DEFINE MEMBER | SET *'member-or-set-name'* AS *'mdx-expression'* ;

UNDEFINE MEMBER | SET *'member-or-set-name' ;*

USER_DEFINED_TRANSLATIONS one or more of 56 locale specifications;

PROC OLAP Statement

The PROC OLAP statement specifies the input data source, cube name, and path. This statement can also be used to

- specify options that might improve query performance
- delete cubes
- specify global settings for handling missing hierarchy members in ragged and unbalanced hierarchies.

PROC OLAP <*option(s)*>;

Options

Note: For information about options that can be used to optimize cube creation and query performance, see "Options Used for Performance" on page 116. △

COMPACT_NWAY
specifies that the cube build will include an additional summarization step that is designed to decrease the size of the NWAY aggregation and improve viewing performance. The amount of improvement depends on the nature of the data. The cubes that improve the most are those that have the largest number of rows that can be included in the additional summarization step.

Candidates for compaction are cubes that are built from star schema, where the cube does not define levels for all columns in all dimension tables. This can result in a fact table that contains many rows that belong to the same leaf member of a given hierarchy. These are the rows that are summarized to decrease the size of the NWAY aggregation.

For example, assume that a cube is built from a star schema that contains a Time dimension table. The Time table contains columns for year, quarter, month, and day, along with a primary key column. If the cube is defined so that the day column is not specified as a level of a time hierarchy, then there are up to 31 key values that refer to each unique combination of year, quarter, and month. Together, these key values define a unique leaf member of that hierarchy. These are the values that are summarized at build time.

The amount of compaction in the NWAY aggregation is determined by the number of source rows that can be summarized. The number of summarized rows depends on the number of unique key values in the fact table that refer to the same leaf member of a hierarchy. Another compaction factor is the number of rows in the fact table that contain unique combinations of keys; these rows are not compacted.

COMPRESS | NOCOMPRESS
specifies whether or not to store the aggregation tables in a compressed format on disk.

Note: This option applies to the automatically created NWAY and all aggregations that do not explicitly specify a COMPRESS option in the AGGREGATION statement. △

Default: NOCOMPRESS

CONCURRENT=*n*
specifies the maximum number of aggregations to create in parallel. This option does not apply to the NWAY aggregation, which is always built first (unless the NO_NWAY option is set).

Default: 2, which is based on the results of a special algorithm that takes into consideration the number of aggregations that are being created and the

number of processors that are available. The algorithm assumes that CPU resources should be reserved for creating aggregation indexes.

Tip: So that each built index has a fair share of the assigned INDEXSORTSIZE memory, INDEXSORTSIZE is divided by the CONCURRENT value. The value of INDEXSORTSIZE should give each concurrent index build enough memory to at least hold a table PARTSIZE. For best performance, INDEXSORTSIZE divided by CONCURRENT should be greater than PARTSIZE.

CUBE=*cube-name*
specifies a valid SAS name for the cube to be created or updated. For naming guidelines, see "Naming Guidelines for SAS OLAP Server" on page 108.

DATA | FACT=*dsname*
specifies the data source for the cube. The unsummarized data source can be any SAS data file, including files that are supported by SAS access engines. If you load the cube from a star schema, then the *dsname* is the name of the fact table that contains the analysis variables from which to derive the measures for the cube. The fact table must also contain fact keys that correspond to dimension tables in the star schema.

You can also provide data set options along with DATA | FACT=. Options are stored within the cube and reapplied when the data is accessed at run time. For more information, see "Data Set Options" in *SAS Language Reference: Concepts*.

Note: This option is not required if you want to define the cube by using input data from a fully summarized external data source (a crossing of all dimensions known as an NWAY); in that case, you specify the data source for the cube by using the TABLE= option in the AGGREGATION statement. △

Interaction: If you load the cube from a star schema, then you must use the DIMENSION statement to

specify the dimension table name (the DIMTBL= option)

specify the dimension (primary) key column (the DIMKEY= option)

specify the column (foreign key) in the fact table that corresponds to the dimension key column (the FACTKEY= option).

DATAPATH=(*'pathname' ...'pathnameN'*)
specifies the location of one or more partitions in which to place aggregation table data. The data is distributed by cycling through each partition location according to the partition size (set using the PARTSIZE= option). For example, if you specify **`DATAPATH=('c:\data1' 'd:\data2')`**, then PROC OLAP places the first partition of each aggregation table into directory c:\data1, the second partition of each table into directory d:\data2, the third partition of each table into c:\data1, and so on. It is also possible to have aggregation tables that use less than the specified number of partitions. For example, your cube might contain an aggregation table that fits entirely into c:\data1.

Note: This option applies to the automatically created NWAY and all aggregations that do not explicitly specify a DATAPATH= option in the AGGREGATION statement. △

Default: The cube subdirectory of the location that is specified by the PATH= option in the PROC OLAP statement

DELETE
deletes the physical cube that is specified with the CUBE= option. It also deletes the cube's definition, which is stored in the metadata repository.

If either the physical cube, its registration, or both are not present, then the DELETE option behaves as explained in the following table:

Note: The DELETE option should only be used if you are recreating a cube from a completely different dataset or table. The use of the DELETE option will remove all information about a cube including security information and information maps. △

Table A1.1 How the DELETE Option Behaves If the Physical Cube or Its Registration Is Not Present

Physical cube exists	Registration exists	DELETE option behaves this way
No	Yes	The physical cube is not deleted. The registration is deleted. If there is a registration, and you use the DELETE option, the registration is always deleted and you cannot recreate the cube from the registration. You can only recreate the cube from the registration when you use the DELETE_PHYSICAL option.
No	No	Fails because there is nothing to delete.
Yes	No	Fails because the cube cannot be located without its registration information. You must manually delete the cube.

DELETE_PHYSICAL
deletes the physical cube that is specified with the CUBE= option but leaves the cube definition intact. This enables you to build a new cube based on the saved cube definition.

If either the physical cube or its registration, or both are not present, then the DELETE_PHYSICAL option behaves as explained in the following table:

Table A1.2 How the DELETE_PHYSICAL Option Behaves If the Physical Cube or Its registration Is Not Present

Physical cube exists	Registration exists	DELETE_PHYSICAL option behaves this way
No	Yes	Fails because there is no physical cube to delete.
No	No	Fails because there is nothing to delete.
Yes	No	Fails because the cube cannot be located without its registration information. You must manually delete the cube.

DESC | DESCRIPTION=*'cube-description'*
specifies any number of characters to be stored as descriptive text. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

DRILLTHROUGH_TABLE | DT_TABLE | DT_TBL=*table-name*
specifies an optional drill-through table. Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source. You can specify the DATA | FACT= table or a different table that includes the necessary data and columns.

You can also specify data set options with this option. Options are stored within the cube and reapplied when the data is accessed at run time. For more information, see "Data Set Options" in *SAS Language Reference: Concepts*.

EMPTY_CHAR='*string*'
specifies the text string that identifies members of character levels that are to be skipped or disregarded. Members are skipped in order to create ragged or unbalanced hierarchies, as described in "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

To be skipped, a member in a character level must have a caption whose value matches the value of the EMPTY_CHAR= option. For example, if a member in a character level is skipped, and if the caption of that member is **Empty**, then the EMPTY_CHAR= option is specified as follows:

```
empty_char='Empty'
```

The maximum length of the quoted string is 256 characters.

Interaction: When specified in the PROC OLAP statement, the EMPTY_CHAR= option can be overridden by the EMPTY_CHAR= or IGNORE_EMPTY options in a HIERARCHY statement or by the EMPTY= or IGNORE_EMPTY options in a LEVEL statement.

To skip members in numeric levels, use the EMPTY_NUM= option.
See also "Naming Guidelines for SAS OLAP Server" on page 108.

EMPTY_NUM='*string*'
specifies the text string that identifies members of numeric levels that are to be skipped or disregarded. Members are skipped in order to create ragged or unbalanced hierarchies, as described in "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

To be skipped, a member in a numeric level must have a caption whose value matches the value of the EMPTY_NUM= option. For example, if a member in a numeric level is skipped, and if the caption of that member is **Empty**, then the EMPTY_NUM= option is specified as follows:

```
empty_num='Empty'
```

The maximum length of the quoted string is 256 characters.

Interaction: When specified in the PROC OLAP statement, the EMPTY_NUM= option can be overridden by the EMPTY_NUM= or IGNORE_EMPTY options in a HIERARCHY statement or by the EMPTY= or IGNORE_EMPTY options in a LEVEL statement.

Note: If there is no format that is associated with the member value, then **BEST12** is used as the format. △

To skip members in character levels, use the EMPTY_CHAR= option.
See also "Naming Guidelines for SAS OLAP Server" on page 108.

IGNORE_MISSING_DIMKEYS=TERSE | VERBOSE
specifying this option when building a cube from a star schema causes SAS to ignore an error condition, log the error, and continue building the cube. The error condition is detected when the fact table contains foreign key values that are not present in one of the contributing dimension tables. By default, and when this option is not specified, any missing dimension keys stop the build of the cube. When IGNORE_MISSING_DIMKEYS=TERSE is specified, the cube build continues and the fact table row with the missing key is ignored (it is not built into the cube). The SAS log receives an entry that lists the total number of key values that are missing from each dimension table. Specifying a value of VERBOSE produces the same behavior, except that the log receives additional details; the missing keys are listed for each dimension table.

INDEX | NOINDEX
specifies whether or not to create the aggregations with indexes. For faster cube creation and adding and deleting aggregations, you can set this option to NOINDEX; however, the lack of indexes will adversely affect query performance.

Note: This option applies to the automatically created NWAY and all aggregations that do not explicitly specify an INDEX option in the AGGREGATION statement. △

Default: INDEX

INDEXPATH=('*pathname*' ...'*pathnameN*')
specifies the locations of the index component files that correspond to each aggregation table partition as specified by the DATAPATH= option.

Note: This option applies to the automatically created NWAY and all aggregations that do not explicitly specify an INDEXPATH= option in the AGGREGATION statement. △

Note: Indexes are not created for aggregations that have fewer than 1,024 records. △

Default: The cube subdirectory of the location that is specified by the PATH= option in the PROC OLAP statement

INDEXSORTSIZE=*n*
specifies the amount of memory in megabytes that is available when aggregations are being created.

Default: The system's available memory

Tip: So that each built index has a fair share of the assigned INDEXSORTSIZE memory, INDEXSORTSIZE is divided by the CONCURRENT value. The value of INDEXSORTSIZE should give each concurrent index build enough memory to at least hold a table PARTSIZE. For best performance, INDEXSORTSIZE divided by CONCURRENT should be greater than PARTSIZE.

MAXTHREADS=*n*
specifies the maximum number of threads that are used to asynchronously create the aggregation indexes. The processing engine calculates how many threads are needed based on the number of indexes that are being created and the INDEXSORTSIZE= value. This option sets a limit on the number of threads regardless of the number that is calculated by the processing engine. However, if the processing engine determines that fewer than the maximum number of threads is needed, then only the calculated number of threads are used.

Default: The value of the SAS system option SPDEMAXTHREADS or 0. If the value is 0, then the processing engine determines the number of threads based on the number of indexes that are created plus the available memory. The maximum value is 65,536 threads.

NO_NWAY
prevents PROC OLAP from automatically creating an NWAY aggregation (the crossing of all dimension levels) for the new cube. The automatically created NWAY is usually the largest in the cube and most resembles the content of the unsummarized data source.

Interaction: If you use this option, then the input data source that is specified with the DATA= or FACT= option must be available at run time; otherwise, queries that are not covered by other aggregations will fail.

PARTSIZE=*partition-size*
specifies the partition size in megabytes of the aggregation table partitions and their corresponding index components.

Note: This option applies to the automatically created NWAY and all aggregations that do not explicitly specify a PARTSIZE= option in the AGGREGATION statement. △

Default: 128 megabytes. The minimum value is 16 megabytes.

PATH=*'pathname'*
specifies the physical or logical path to the location of a new cube. Within the specified path, the cube is stored in a directory that uses the name of the cube in uppercase letters. For example, if you enter the path **'c:\v9cubes'** and the cube name is **MrktData**, then the cube is stored in **'c:\v9cubes\MRKTDATA'**. Enclose the path within quotation marks.

REGISTER_ONLY
specifies that metadata for a cube is to be registered, but the cube is not to be physically built. All of the metadata for the cube is added to the SAS Metadata Repository. The physical cube can be built later using the existing metadata definition, as specified in *Building a Cube from an Existing Definition* in the help for SAS OLAP Cube Studio. Note that all data sets must physically exist at registration time. The data sets can be empty—they do not need to contain data. Complete data sets are required when the cube is physically built.

SEGSIZE=*rows-per-segment*
specifies the number of observations (table rows) in the file segment of the index component. The value is expressed in multiples of 1,024. The minimum value is 1 (1,024 rows). The segmented indexes are used to optimize the processing of WHERE expressions. Each parallel thread is given a segment of the table to evaluate that is equal to the value of the SEGSIZE= option multiplied by 1,024.

Note: This option applies to the NWAY aggregation and all aggregations that do not explicitly specify a SEGSIZE= option in the AGGREGATION statement. △

Default: 8 (8 x 1,024 = 8,192 rows)

SYNCHRONIZE_LEVELS
checks for SAS column name changes and cube level name changes in an existing cube. This option will find the name differences and change the level names to match the column names. When the SYNCHRONIZE_LEVELS option is set, only the METASVR statement is allowed. No other option or statement can be used.

Note: All source and aggregation column names for a level must be the same. △

WORKPATH=(*'pathname1' ...'pathnameN'*)
specifies one or more locations for temporary work files.

Default: For all operating environments except z/OS and VMS, if the WORKPATH= option is not specified, PROC OLAP uses the SPDEUTILLOC= system option. If SPDEUTILLOC= is not specified, PROC OLAP uses the UTILLOC= system option. If UTILLOC= is not specified, or if you do not have write access to the specified path, the follow message is generated:

```
ERROR: Cannot create temporary index for proc olap.
NOTE: The SAS System stopped processing this step
because of errors.
```

For z/OS and VMS, PROC OLAP uses the SPDEUTILLOC= system option only.

Note: The SPDE options are honored by PROC OLAP only if the REGISTER_ONLY option is set on the PROC OLAP statement, and only if the long form of the procedure is used to build a cube. The long form of the procedure is used when you run a SAS program that contains PROC OLAP code. The short form of the procedure is used by SAS OLAP Cube Studio. △

METASVR Statement

The METASVR statement identifies the SAS metadata repository in which existing cube metadata information exists or in which metadata about a new cube is stored.

METASVR OLAP_SCHEMA='*schema-name*' < *option(s)*>;

The METASVR statement options can be used to override the metadata repository connection values that are specified through SAS start-up options.

Note: During an interactive SAS session, if connection information is not available either through start-up settings or through a METASVR statement, then the user is prompted for the missing information. For more information about SAS start-up options, see *SAS Language Reference: Dictionary*. △

Following is an example of a METASVR statement with all of its options set:

```
metasvr olap_schema='Banking Schema'
   repository='financial repository'
   host='misdept.us.mar.com'
   port=9999
   protocol=bridge
   userid=jjones
   pw='my password';
```

Required Argument

OLAP_SCHEMA='*schema-name*'
is a string that specifies the name of the schema that has been defined in a SAS metadata repository. The name can be a maximum of 32 characters. The OLAP schema specifies which group of cubes that a SAS OLAP Server can access. Each OLAP schema can be accessed by multiple SAS OLAP Servers; however, each SAS OLAP Server has access to only one OLAP schema. When using embedded blanks or special characters in the schema name, enclose the name in quotation marks.

Options

HOST='*metadata-server-host-name*'
is a string that specifies the IP address of the metadata repository host. An example is **'misdept.us.mar.com'**. The address can be a maximum of 256 characters. When using lowercase letters, embedded blanks, or special characters in the host name, enclose the name in quotation marks.

PORT=*port-number*
specifies the numeric value of the port on which the metadata repository resides.

PROTOCOL=BRIDGE | COM
specifies the protocol that is used to connect to the specified metadata repository.

PW='*password*'
is a string that specifies the password for the user identified with the USERID= option. The password can be a maximum of 512 characters. When using lowercase letters, embedded blanks, or special characters in the password, enclose the password in quotation marks.

REPOSITORY=*'repos-name'*
is a string that specifies the name of a SAS metadata repository in which existing cube metadata information exists or in which metadata about a new cube is stored. The name can be a maximum of 60 characters. When using lowercase letters, embedded blanks, or special characters in the repository name, enclose the name in quotation marks.

USERID=*'userid'*
is a string that specifies the user's identification for the specified metadata repository. The identification can be a maximum of 256 characters. When using lowercase letters, embedded blanks, or special characters in the user ID, enclose the user ID in quotation marks.

DIMENSION Statement

The DIMENSION statement defines the logical and hierarchical relationships between the variables in the input data.

DIMENSION *dim-name* HIERARCHIES=(*hier–nam ... hier-nameN*) <*option(s)*>;

At least one DIMENSION statement must be specified when the cube is created. The DIMENSION statement is not used when adding or deleting aggregations from cubes.

The maximum number of dimensions that can be defined in a cube is determined by combining the number of dimensions with the number of multiple hierarchies that are defined in those dimensions. The maximum value of that sum is 128. Mathematically, the sum is expressed as follows:

```
MaxDims = NumDims + NumMultipleHeirarchies = 128
```

All hierarchies other than the first hierarchy in each dimension apply to the total.

Here are some examples of cubes that are defined with the maximum number of dimensions:

128 dimensions, each dimension has 1 hierarchy

127 dimensions, 1 dimension has 2 hierarchies

126 dimensions, 1 dimension has 3 hierarchies

126 dimensions, 2 dimensions have 2 hierarchies

The DIMENSION statement does *not* create aggregations. To create aggregations, use the AGGREGATION statement.

A DIMENSION statement must include the name of at least one hierarchy in its HIERARCHIES= option. In addition, a HIERARCHY statement must include the name of at least one level in its LEVELS= option. Note that you cannot use the same level in more than one dimension.

You can use LEVEL statements to specify a time period for each level in a TIME dimension. You can also use LEVEL statements to supply information such as a level-specific sort order or a level description.

The following example uses one DIMENSION statement, two HIERARCHY statements, and three optional LEVEL statements to define a fully specified dimension. In the example, the same levels are being used in different ways.

```
dimension time
   hierarchies=(Year_Months Year_Quarters)
   ;
   hierarchy Year_Months
      levels=(year month day)
```

```
   ;
hierarchy Year_Quarters
   levels=(year quarter day)
   ;
level year
   type=year
   caption='Year'
   ;
level quarter
   type=quarters
   caption='Quarter'
   ;
level month
   type=months
   caption='Month'
   ;
level day
   type=days
   caption='Day'
   ;
```

Required Arguments

dim-name
names a dimension by using a valid SAS name up to 32 characters. For naming guidelines, see "Naming Guidelines for SAS OLAP Server" on page 108.

HIERARCHIES=*(hier-name...hier-nameN)*
specifies the name of one or more hierarchies as defined by HIERARCHY statements.

Options

CAPTION=*'string'*
specifies a maximum of 256 characters that can be used to create a meaningful description of the dimension. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: *dim-name*

DESC | DESCRIPTION=*'string'*
specifies any number of characters that can be used to create a meaningful description of the dimension. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: *dim-name*

DIMKEY=*dimension-table-column*
specifies the name of the column in the dimension table that is specified in the DIMTBL= option. That column must contain values that correspond to fact key values in the fact table and be a value that corresponds to a unique combination of level values in the fact table.

Note: The corresponding fact key is specified with the FACTKEY= option. The fact table is specified with the FACT= option in the PROC OLAP statement. △

For example, for a dimension that is composed of three levels—**NAME**, **ADDRESS**, and **INCOME**—a dimension key named **CUSTOMER_ID** might exist. In this dimension, each unique value of **CUSTOMER_ID** corresponds to a unique combination of **NAME**, **ADDRESS**, and **INCOME**.

Table A1.3 Sample Dimension Data That Illustrates How Unique DIMKEY Values Correspond to Unique Combinations of Level Values

CUSTOMER_ID	NAME	ADDRESS	INCOME
1	Juan	hostel	2000
2	Shelly	apartment	2000
3	Paul	house	25000
4	Makoto	castle	250000000

DIMTBL=*libname.memname*
specifies the valid, two-level SAS name for a dimension table in the star schema that is specified with the FACT= option in the PROC OLAP statement. The dimension table must contain one column for each dimension level name (specified with the LEVELS= option in HIERARCHY statements) and one column for the dimension key. However, if the dimension key is also a level, then the dimension table needs to have only as many columns as there are levels in the dimension. Member metadata for the dimension is derived from the information in the level columns of the dimension table.

You can also specify data set options with DIMTBL=. Options are stored within the cube and reapplied when the data is accessed at run time. For more information, see "Data Set Options" in *SAS Language Reference: Concepts*.

Note: The fact table does not have to contain all of the members. However, the fact table cannot contain any members that are not described by the level columns. △

Note: The same dimension tables can be used to load cubes that have some, but not all, dimensions in common. This means that it is possible for multiple cubes to share the same dimension *data*. △

Note: If you are building a cube that will contain multiple national languages, then replace the DIMTBL= option with DIMTABLELIBREF= and DIMTABLEMEMPREF= options. In addition, you must create a USER_DEFINED_TRANSLATIONS statement. △

DIMTABLELIBREF=
specifies the library for the data sets that exist, for this dimension, in each language that is specified by the USER_DEFINED_TRANSLATIONS statement. The library is associated with the dimension and not the language. You cannot put different languages in different libraries, but you can put different dimensions in different libraries. This option is required if you are using the Multiple Language Support capabilities of the SAS OLAP Server. It is also used in conjunction with the DIMTABLEMEMPREF= option.

Note: If you are building a cube that will contain multiple national languages, then DIMTABLELIBREF= and DIMTABLEMEMPREF= are required instead of DIMTBL=. △

DIMTABLEMEMPREF=
specifies the member prefix for the translated dimension tables. The member prefix is the prefix of the data set name. The suffix of the name is provided by the USER_DEFINED_TRANSLATIONS statement. For example, if the member prefix is **`dealdim_`** and the suffix is **`da_DK`**, then PROC OLAP looks for a data set named **`dealdim_da_DK.sas7bdat`** in the library that is specified by the DIMTABLELIBREF= option. DIMTABLEMEMPREF= is required if you are using the Multiple Language Support capabilities of the SAS OLAP Server. It is used in conjunction with the DIMTABLELIBREF= option and the USER_DEFINED_TRANSLATIONS statement. This option follows the "Naming Guidelines for SAS OLAP Server" on page 108.

Note: If you are building a cube that will contain multiple national languages, then DIMTABLELIBREF= and DIMTABLEMEMPREF= are required instead of DIMTBL=. △

FACTKEY=*fact-table-column*
specifies the name of the column in the fact table that corresponds to the dimension table column that is specified with the DIMKEY= option. The name does not have to match the DIMKEY name. Referring back to the previously discussed example, the FACTKEY name could be **`CUST_NO`** even though the DIMKEY name is **`CUSTOMER_ID`**. However, even if the names are different, the underlying data must match. For example, you must match numeric columns with numeric columns and character columns with character columns. In addition, if the FACTKEY is a character column, then it must be the same length as the DIMKEY column. If the FACTKEY is a numeric column, then it is handled as a decimal precision number (rather than as an integer).

SORT_ORDER=ASCENDING | DESCENDING | ASCFORMATTED | DESFORMATTED | DSORDER
specifies a sort order for all levels in the dimension. Values that are returned from queries display in this order by default.

Default: ASCENDING

Interaction: This setting is overridden if sort order is set in a LEVEL statement.

Tip: To specify a sort order for each level within a dimension, set the SORT_ORDER= option in each LEVEL statement. Values that are returned from queries display in this order.

Note: The sort order can be changed at query time using the MDX ORDER functions.

TYPE=TIME | GEO
identifies the dimension as a TIME or GEO dimension. The GEO type is used when defining ESRI map information for a cube.

Requirement: You must set this option for a TIME or GEO dimension. TIME and GEO are the only valid values for this option.

Interaction: You can use LEVEL statements to specify the time period of each level in the TIME dimension. Specifying TYPE=TIME also allows you to use the MDX time series functions during data query.

LEVEL Statement

The LEVEL statement provides additional information about a level specified with the LEVELS= option in a HIERARCHY statement, and enables you to set options for ragged hierarchies.

LEVEL *level-name* <*option(s)*>;

For TIME dimensions, you can use LEVEL statements to specify a time period for each level in the dimension. However, if you specify the time period for one level, then you must specify the time period for all levels. You also use LEVEL statements to supply information such as a level description or a level-specific sort order. You can have a maximum of 256 levels per cube and a maximum of 19 levels per hierarchy.

Note: Levels that are shared between hierarchies share the values of the options EMPTY_CHAR=, EMPTY_NUM=, EMPTY=, and IGNORE_EMPTY. These options are used to create ragged or unbalanced heirarchies, as described in "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17. △

Note: Levels use formats as specified in the input data source. To override the format, you can use a SAS FORMAT statement. △

Note: When you rebuild a cube that has been physically deleted, the rebuilt cube still uses the formats that were originally used to build the cube and were saved in the cube's metadata. This means that the rebuilt cube does not automatically include any formatting changes that you might have made in the input data source. To manually specify the new formats, edit and rebuild the cube by using SAS OLAP Cube Studio. △

Required Arguments

level-name
specifies a valid SAS name for the level that matches the name of a corresponding column in the input data. (You can use a column as a level even if it is also being used as a measure.) This is the same name that is used in the LEVELS= option in the HIERARCHY statement. Level names must be unique within a cube. For naming guidelines, see "Naming Guidelines for SAS OLAP Server" on page 108.

Options

CAPTION='*string*'
specifies a maximum of 256 characters that can be used to create a meaningful description of the level. Third-party applications that report on cube data might display this description. If the text includes blank spaces or special characters that are not permitted in a valid SAS name, then enclose the caption within quotation marks.

Default: The column's label in the input data source. If there is no label available, the default is the level name.

DESC | DESCRIPTION='*string*'
specifies any number of characters that can be used to create a meaningful description of the level. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: The value of the CAPTION= option if one exists; otherwise, the column's label. If there is no label available, the default is the level name.

EMPTY='*string*'
specifies the text string that identifies members that are to be skipped or disregarded. Members are skipped in order to create ragged or unbalanced hierarchies, as described in "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

To be skipped, a member must have a caption whose value matches the value of the EMPTY= option. For example, if a member is skipped, and if the caption of that member is **Empty**, then the EMPTY= option is specified as follows:

```
empty='Empty'
```

The maximum length of the quoted string is 256 characters.

Interaction: The EMPTY= option overrides for that level any specification of EMPTY_CHAR=, EMPTY_NUM=, or IGNORE_EMPTY that might have been specified in the respective HIERARCHY or PROC OLAP statement.

See also "Naming Guidelines for SAS OLAP Server" on page 108.

IGNORE_EMPTY
specifies that any value of the EMPTY_CHAR= option (for character levels) or EMPTY_NUM= option (for numeric levels) that was specified in the respective HIERARCHY or PROC OLAP statement is to be ignored. The level is not to be skipped in the cube build. For further information, see "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

SORT_ORDER=ASCENDING | DESCENDING | ASCFORMATTED | DESFORMATTED | DSORDER
specifies a sort order for a level within a dimension. Values that are returned from queries display in this order.

Default: If a sort order is not specified in the DIMENSION statement or in the LEVEL statement, then the default order of ASCENDING is applied.

Interaction: This setting overrides the SORT_ORDER= setting in the DIMENSION statement.

TYPE=YEAR | HALF_YEARS | QUARTERS | MONTHS | WEEKS | DAYS | HOURS | MINUTES | SECONDS
if you specify the TYPE=TIME option in the DIMENSION statement, then you can use this LEVEL statement option to specify the time period for the dimension levels.

Requirement: If you specify a time period for one level in the TIME dimension, then you must specify the time period for all levels in the dimension. With regard to drill path, identify the levels from the most general time period to the most specific.

PROPERTY Statement

The PROPERTY statement assigns properties to specific levels within specified hierarchies.

PROPERTY *prop-name* LEVEL=*level-name* <*option(s)*>;

Each level can have more than one property assigned to it by using multiple PROPERTY statements. Property names must match the name of a column in the input data source, or you must use the COLUMN= option to specify the column name.

In the following example, the COLUMN= option is used in the first two PROPERTY statements because the column name is different from the property name. In this way, the property named **Population** can be assigned to both the **country** level and the **state** level in the **geo** hierarchy. The level **state** has two properties: **Population** and **West_of_Miss**.

```
property Population
   column=p_country
```

```
   hierarchy=geo
   level=country
   ;
property Population
   column=p_state
   hierarchy=geo
   level=state
   ;
property West_of_Miss
   hierarchy=geo
   level=state
   ;
```

Required Arguments

prop-name
specifies a valid SAS name for the property. Usually this is the name of a column in the input data source. If it is not the name of a column, then you must include the COLUMN= option to specify the column name. For naming guidelines, see "Naming Guidelines for SAS OLAP Server" on page 108.

LEVEL=*level-name*
specifies the name of the level that you are assigning the property to.

Options

CAPTION=*'string'*
specifies a maximum of 256 characters that can be used to create a meaningful description of the level. Third-party applications that report on cube data might display this description. If the caption includes blank spaces or special characters that are not permitted in a valid SAS name, then enclose the caption within quotation marks.

Default: The column's label if it exists, otherwise the property name.

COLUMN=column-name
specifies the name of a column from the input data source. You must use this option if the column name is not the same as the property name.

DESC | DESCRIPTION=*'string'*
specifies any number of characters that can be used to create a meaningful description of the level. Third-party applications that report on cube data might display this description. If the description includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: The value of the CAPTION= option if one exists; otherwise, the column's label.

HIERARCHY=(*hier-name ... hier-nameN*)
specifies the name of one or more hierarchies that contain the level. If you do not include the HIERARCHY option, then the property is automatically assigned to all occurrences of the level in all of the hierarchies in which it appears; otherwise, the property is assigned to the level only in the specified hierarchies.

HIERARCHY Statement

The HIERARCHY statement specifies the navigational order of the levels in a dimension.

HIERARCHY *hier-name* LEVELS=(*level–name1 <level–name2 ...level-nameN>*) *<option(s)>*;

You must define at least one hierarchy for each dimension. Specifically, each DIMENSION statement must identify at least one unique HIERARCHY statement.

The maximum number of hierarchies that can be defined in a dimension is determined only by the maximum number of hierarchies that can be defined in a cube. The maximum number of hierarchies that can be defined in a cube is determined by combining the number of multiple hierarchies with the number of dimensions. The maximum value of that sum is 128. Mathematically, the sum is expressed as follows:

```
MaxHiers = NumMultHiers + NumDimensions = 128
```

All hierarchies other than the first hierarchy in each dimension apply to the total.

Here are some examples of cubes that meet the maximum number of hierarchies:

128 dimensions, each dimension has 1 hierarchy

127 dimensions, 1 dimension has 2 hierarchies

126 dimensions, 1 dimension has 3 hierarchies

126 dimensions, 2 dimensions have 2 hierarchies

Levels in the same dimension can be shared between hierarchies. Every level in a dimension must be assigned to a hierarchy in the dimension. You can have a maximum of 19 levels per hierarchy. There is no limit to the number of hierarchies per dimension.

Following is an example of a HIERARCHY statement that specifies three levels:

```
hierarchy Geography
   levels=(country region division);
```

Required Arguments

hier-name
specifies a valid SAS name for the hierarchy. This name is also used in the HIERARCHIES= option in the DIMENSION statement. The *hier-name* cannot be the same as any of its level names. Hierarchy names must be unique within the cube. If the hierarchy that you are defining is the only one in the dimension, then the hierarchy name must match the dimension name. For other naming guidelines, see “Naming Guidelines for SAS OLAP Server” on page 108.

LEVELS=(*level-name1 <level-name2 ...level-nameN>*)
specifies a valid SAS name for at least one level. These names correspond to columns in your input data and are used in any optional LEVEL statements. Level names must be unique within a cube and cannot be the same as the *hier-name*. (You can use a column as a level even if it is also being used as a measure.) Enter one or more names, separated by a space. Enter the level names in the order in which you want them to be used, beginning with the top level. For naming guidelines, see “Naming Guidelines for SAS OLAP Server” on page 108.

Requirement: If the hierarchy is part of a TIME dimension, then the levels must be listed in order from most general to least general based on their assigned TYPE. For example, a TYPE=YEAR level must be listed before a TYPE=QUARTER level.

Options

CAPTION='*string*'
specifies a maximum of 256 characters that can be used to create a meaningful description of the hierarchy. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: *hier-name*

DEFAULT
identifies a hierarchy as the default hierarchy for the dimension that is defined by the DIMENSION statement.

Default: The first hierarchy listed for the dimension

DESC | DESCRIPTION='*string*'
specifies any number of characters that can be used to create a meaningful description of the hierarchy. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: *the hierarchy caption, which may be the default, hier-name*.

EMPTY_CHAR='*string*'
specifies the text string that identifies members of character levels that are to be skipped or disregarded. Members are skipped in order to create ragged or unbalanced hierarchies, as described in "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

To be skipped, a member in a character level must have a caption whose value matches the value of the EMPTY_CHAR= option. For example, if a member in a character level is skipped, and if the caption of that member is **Empty**, then the EMPTY_CHAR= option is specified as follows:

```
empty_char='Empty'
```

The maximum length of the quoted string is 256 characters.

Interaction: When specified in the HIERARCHY statement, the EMPTY_CHAR= option overrides (for that hierarchy) any specification of the EMPTY_CHAR= option in the PROC OLAP statement. In turn, the EMPTY_CHAR= option in the HIERARCHY statement is overridden by the EMPTY= or IGNORE_EMPTY options in the LEVEL statements in that hierarchy.

To skip members in numeric levels, use the EMPTY_NUM= option.
See also "Naming Guidelines for SAS OLAP Server" on page 108.

EMPTY_NUM='*string*'
specifies the text string that identifies members of numeric levels that are to be skipped or disregarded. Members are skipped in order to create ragged or unbalanced hierarchies, as described in "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

To be skipped, a member in a numeric level must have a caption whose value matches the value of the EMPTY_NUM= option. For example, if a member in a numeric level is skipped, and if the caption of that member is **Empty**, then the EMPTY_NUM= option is specified as follows:

```
empty_num='Empty'
```

The maximum length of the quoted string is 256 characters.

Interaction: When specified in the HIERARCHY statement, the EMPTY_NUM= option overrides (for that hierarchy) any specification of the EMPTY_NUM= option in the PROC OLAP statement. In turn, the EMPTY_NUM= option in the HIERARCHY statement is overridden by the EMPTY= or IGNORE_EMPTY options in the LEVEL statements in that hierarchy.

Note: If there is no format that is associated with the member value, then **BEST12** is used as the format. △

To skip members in character levels, use the EMPTY_CHAR= option.

See also "Naming Guidelines for SAS OLAP Server" on page 108.

IGNORE_EMPTY

specifies that, for this hierarchy, any values that were specified for the EMPTY_CHAR= and EMPTY_NUM= options in the PROC OLAP statement are to be ignored. This option can be overridden by specifications of EMPTY_CHAR= and EMPTY_NUM= in the same HIERARCHY statement. The IGNORE_EMPTY option can also be overridden in subsequent LEVEL statements using the EMPTY= option. For further information, see "Defining Ragged and Unbalanced Hierarchies with PROC OLAP" on page 17.

MEASURE Statement

The MEASURE statement defines the cube's measures and indicates how they map to the input data.

MEASURE *measure-name* STAT=*statname* <*option(s)*>;

Include one MEASURE statement for each measure in the cube. Each cube must have at least one measure. Measure names must be unique. You can have a maximum of 1,024 measures per cube.

Note: All cube aggregations have identical measures. △

Required Arguments

measure-name

specifies a valid SAS name for the measure. The name must be unique. For naming guidelines, see "Naming Guidelines for SAS OLAP Server" on page 108.

STAT= *statname*

specifies the statistic for the measure. The following base statistics are available: N, NMISS, NUNIQUE, SUM, MAX, MIN, or USS. In addition, these derived statistics are also available: AVG, RANGE, CSS, VAR, STD, STDERR, CV, T, PRT, LCLM, or UCLM.

When specifying the NUNIQUE statistic with the MDXfunction DISTINCTCOUNT, it will return empty data when there is not a valid crossing of data. This can be removed when the axis uses non-empty. In addition, the NUNIQUE statistic will not be supported for cubes that contain ragged or unballanced hierarchies.

Note: At least one non-NUNIQUE measure must be defined. △

New cubes that are based on a data source that contains *existing summarized data* (where such data has been indicated in at least one AGGREGATION statement via the TABLE= option), must include measure statements for the

stored statistics required for each derived statistic that you want to create for the new cube. For example, if you want to calculate AVG, you must create measures for N and SUM, as well as AVG.

Note: MOLAP aggregations do not require the N and SUM. △

The following table indicates which stored statistics are required for each derived statistic:

Table A1.4 Stored Statistics Required for Each Derived Statistic

Derived Statistics	Required Stored Statistics
AVG	N, SUM
CSS	N, SUM, USS
RANGE	MIN, MAX
VAR, STD, STDERR, CV, T, PRT, LCLM, UCLM	N, SUM, USS

Note: For information about statistic formulas, see "Keywords and Formulas" in *Base SAS Procedures Guide*. △

For cubes that are *not* loaded from a fully summarized data source (that is, you specified a data source by using the DATA | FACT= option), some statistics use formats taken from the input data source. Specifically, if the statistic is SUM, MIN, MAX, RANGE, AVG, STD, STDERR, LCLM, or UCLM, then PROC OLAP uses the format that is assigned to the column specified by the COLUMN | ANALYSIS= option. The following table lists the formats used for the other supported statistics:

Table A1.5 Default Formats Used for Statistics

Statistic	Format Used
CSS	BEST.
CV	8.2
N	12.0
NMISS	10.0
PRT	6.4
T	7.3
USS	BEST.
VAR	BEST.

For cubes that *are* loaded from a fully summarized data source (that is, you specified the data source by using the AGGREGATION statement), the default format is BEST12.

To override the default formats, you can either set the FORMAT= option or use a SAS FORMAT statement.

Note: The FORMAT= option also overrides a FORMAT statement. △

Note: When you rebuild a cube that has been physically deleted, the rebuilt cube still uses the formats originally saved in the cube's metadata. This means that the rebuilt cube does not automatically include any formatting changes that you might have made in the input data source. To manually specify the new formats, edit and rebuild the cube by using SAS OLAP Cube Studio. △

COLUMN | ANALYSIS=*anlvar*
specifies the name of a numeric column that is contained in the cube's input data source. (You can use a column as a measure even if it is also being used as a level.)

If the cube is based on an *unsummarized* data source, then *anlvar* is the name of the column in that data source from which the measure will be calculated. Use COLUMN= to specify the column.

If the cube is based on a *summarized* data source, then *anlvar* can be the name of the numeric column in the data source that was used as the analysis variable for the pre-calculated measure. Use ANALYSIS= to specify the column. It can also be a name that identifies a logical association between measures with the same *anlvar* name.

For example, if your cube has three measures, N, SUM, and AVERAGE, and if those measures were derived from the same analysis variable, then you could specify **`ANALYSIS=Sales`** to logically link the three measures through their shared analysis variable. You would also identify the analysis variable in the AGGR_COLUMN= option.

As a further illustration, assume that you were building a cube with an NWAY aggregation that was specified using a summarized SAS dataset. The dataset contains the columns Country, Region, Division, Year, Quarter, Month, SumOfSales, and NumOfSales. You would use two MEASURES statements, one for SumOfSales and another for NumOfSales, as follows.

```
measure Sales_Sum
   stat=sum
   aggr_column="SumOfSales"
   analysis="Sales"
   desc='Sum of Sales'
   units='Dollars'
   format=dollar10.2
   ;
measure Sales_N
   stat=n
   aggr_column="NumOfSales"
   analysis="Sales"
   desc='Number of Sales'
   units='Dollars'
   format=dollar10.2
   ;
```

The Sales column becomes logically linked with the physical columns SumOfSales and NumOfSales.

If the cube consists of a combination of summarized and unsummarized data sources, then *anlvar* refers to both a physical and a logical entity. For example, you might have a cube that requires a physical analysis variable to create a crossing but that same cube already contains other, higher level aggregations. In this case, the analysis variable is also used to logically link the measures in the pre-existing aggregations that were derived from the same input column. You would also identify the analysis variable in the AGGR_COLUMN= option.

Default: *measure-name*

Interaction: An unsummarized data source is specified with the DATA | FACT= option in the PROC OLAP statement. A summarized data source is specified with the TABLE= option in an AGGREGATION statement.

Note: The COLUMN argument is not required for the NUNIQUE statistic and will be ignored for the NUNIQUE statistic if specified. △

Options

AGGR_COLUMN=*input-column*
specifies the name of the numeric column in the summarized input data that contains the values for the measure. The source of the summarized input data is specified in the AGGREGATION statement. This option is valid only for stored statistics.

Default: *measure-name*

CAPTION=*'string'*
specifies a maximum of 256 characters that can be used to create a meaningful description of the measure. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: The default is based on the statistic and the COLUMN= value, as shown in the following table. For example, if the statistic is **`SUM`** and the COLUMN= value is **`Sales`**, then the default caption is **`Sum of Sales`**.

Table A1.6 Defaults for the CAPTION= Option If No Caption Is Specified

Statistic Used for Measure	Default Caption
AVG	Average *measure-column-name*
CSS	Corrected Sum of Squares of *measure-column-name*
CV	*Measure-column-name* Coefficient of Variation
LCLM	*Measure-column-name* Lower Confidence Limit
MAX	Maximum *measure-column-name*
MIN	Minimum *measure-column-name*
N	Number of Values for *measure-column-name*
NMISS	Number of Missing Values for *measure-column-name*
NUNIQUE	Number of Unique Values for level-name in hierarchy-name
PRT	Probability of Greater Absolute Value for *measure-column-name*
RANGE	*Measure-column-name* Range
STD	*Measure-column-name* Standard Deviation
STDERR	*Measure-column-name* Standard Error of Mean
SUM	Sum of *measure-column-name*
T	*Measure-column-name* T Value
UCLM	*Measure-column-name* Upper Confidence Limit

Statistic Used for Measure	Default Caption
USS	*Measure-column-name* Uncorrected Sum of Squares
VAR	*Measure-column-name* Variance

DEFAULT
identifies a measure as the default measure for the cube.

Default: The measure defined in the first MEASURE statement

DESC | DESCRIPTION=*'string'*
specifies any number of characters that can be used to create a meaningful description of the measure. Third-party applications that report on cube data might display this description. If the text includes blank spaces or any characters that are not permitted in a valid SAS name, then enclose the text within quotation marks.

Default: *measure-name*

FORMAT=*sas-format-name*
specifies the SAS format to be used to display the value of the measure. This format overrides the default format (see STAT= for more information) and any format that is specified in a SAS FORMAT statement.

Note: When you rebuild a cube that has been physically deleted, the rebuilt cube still uses the formats that were originally saved in the cube's metadata. This means that the rebuilt cube does not automatically include any formatting changes that you might have made in the input data source. To manually specify the new formats, edit and rebuild the cube by using SAS OLAP Cube Studio. △

HIERARCHY=*'string'*
specifies the hierarchy in which the level resides. This option is used only with the NUNIQUE statistic. If there is only one hierarchy, then the option may be omitted.

Note: The HIERARCHY= option will be ignored for non-NUNIQUE statistics if specified. △

LEVEL= *'string'*
specifies the level for which a unique count will be determined. This option is used only with the NUNIQUE statistic. The default is the Measure name.

Note: The LEVEL= option will be ignored for non-NUNIQUE statistics if specified. △

UNITS=*'string'*
specifies a maximum of 256 characters that can be used to create a meaningful description of the measure's units (for example, "pounds sterling"). Third-party applications that report on cube data might display this description. If the text includes blank spaces, mixed-case letters, special characters, then enclose the text within quotation marks.

AGGREGATION Statement

The AGGREGATION statement defines an aggregation of the cube based on level information that you provide.

AGGREGATION level-name <*level-name2 level-name3 ...level-nameN*> / <*option(s)*>;

You can specify level names that are associated with an unsummarized data source, or you can specify level names that match columns in a table that contains existing aggregated data. The levels can exist in more than one dimension. You do not need to include dimension names because level names must be unique across dimensions.

Here is an example of an AGGREGATION statement that specifies three levels and uses the NAME= option. The slash character (/)is required to separate level names from option specifications.

```
aggregation country prodtype year /
   name='Product Types by Country';
```

Required Arguments

level-name
is the level that is to be used to create the aggregation. Additional level names are optional. Names are separated by spaces. Names are separated from option specifications with a required slash character (/). You do not have to include all levels that are specified in all HIERARCHY statements, but the names that you do specify must match the names that are used in the HIERARCHY statements. You can include a TABLE= option to identify a table that contains existing aggregated information for your specified levels. The levels that you specify must match columns in the input table.

Restriction: Levels must be listed in drill-path order. You cannot specify an aggregation that contains a summary level that could never be requested. For example, if your TIME hierarchy contains the levels **`Year`**, **`Month`**, and **`Day`**, you could specify **`Year`** and **`Month`** as an aggregation but not **`Month`** by itself.

Options

Note: For information about options that can be used to optimize cube creation and query performance, see "Options Used for Performance" on page 116Syntax Options Used for Performance. △

COMPRESS | NOCOMPRESS
specifies whether or not to store the aggregation table in a compressed format on disk.

Default: NOCOMPRESS

DATAPATH=(*'pathname' ...'pathnameN'*)
specifies the location of one or more partitions in which to place aggregation table data. The data is distributed by cycling through each partition location according to the partition size. This is set by using the PARTSIZE= option. For example, if you specify **`DATAPATH=('c:\data1' 'd:\data2')`**, then PROC OLAP places the first partition of the aggregation table into directory **`c:\data1`**, the second partition of the table into directory **`d:\data2`**, the third partition of the table into **`c:\data1`**, and so on. It is also possible to have aggregation tables that use fewer than the specified number of partitions. For example, your aggregation table might fit entirely into **`c:\data1`**.

Default: The cube subdirectory of the location that is specified by the PATH= option in the PROC OLAP statement

INDEX | NOINDEX
specifies whether or not to create the specified aggregation with indexes. For faster cube creation and updates, you can set this option to NOINDEX; however, the lack of indexes might adversely affect query performance.

Note: Indexes are not created for aggregations that have fewer than 1,024 records. △

Default: INDEX

INDEXPATH=(*'pathname' ...'pathnameN'*)
specifies the locations of the index component files that correspond to each aggregation table partition as specified by the DATAPATH= option.

Default: The cube subdirectory of the location that is specified by the PATH= option in the PROC OLAP statement

NAME=*'aggregation-name'*
specifies a maximum of 256 characters as the name of the aggregation. If the name includes blank spaces or any characters that are not permitted in a valid SAS name, then the name must be enclosed within quotation marks. The name is stored with the cube's metadata.

Default: A name assigned by SAS such as **AGGR1**.

PARTSIZE=*partition-size*
specifies the partition size in megabytes of the aggregation table partitions and their corresponding index components.

Default: 128 megabytes. The minimum value is 16.

SEGSIZE=*rows-per-segment*
specifies the number of observations (table rows) in the file segment of the index component. The value is expressed in multiples of 1,024. The minimum value is 1 (1,024 rows). The segmented indexes are used to optimize the processing of WHERE expressions. Each parallel thread is given a segment of the table to evaluate that is equal to the value of the SEGSIZE= option multiplied by 1,024.

Default: 8 (8 x 1,024 = 8,192 rows)

Interaction: The value of this option overrides for the current aggregation any such value that was specified for all aggregations in the PROC OLAP statement.

TABLE=*libname.dataset*
specifies the name of a SAS data set or data view that contains the data for one aggregation. Every level that is listed in the AGGREGATION statement must match a column that contains aggregation information in the specified table. Place this option after the list of level names.

Analysis columns in the table are mapped to the numeric columns that are specified with the AGGR_COLUMN= option in MEASURE statements.

You can also set data set options with TABLE=. Options are stored within the cube and reapplied when the data is accessed at run time. For more information, see "Data Set Options" in *SAS Language Reference: Concepts*.

Restriction: You cannot use the TABLE= option in an AGGREGATION statement that is used to add an aggregation to an existing cube.

DROP_AGGREGATION Statement

The DROP_AGGREGATION statement removes an aggregation from the specified cube.

DROP_AGGREGATION *level-name1 < level-name2 ...level-nameN>* **/**
NAME=*aggregation-name*;

You can specify the levels that are in the aggregation, or the name of the aggregation, or both the levels and the name. The slash character (/)is required to separate level names from option specifications.

Required Arguments

At least one of the following arguments is required for a DROP_AGGREGATION statement:

level-name1
specifies the names of the level that is in the aggregation that you want to drop. Additional levels can be specified using blank spaces to separate the level names, as shown in the following example.

```
drop_aggregation Year Month Product /
   name=Sales ;
```

NAME=*'aggregation-name'*
specifies the name of the aggregation that you want to drop. If the name includes blank spaces or any characters that are not permitted in a valid SAS name, then the name must be enclosed within quotation marks.

DEFINE Statement

The DEFINE statement defines a global calculated member or a named set for any cube that is registered in a SAS metadata repository.

DEFINE MEMBER | SET *'member-or-set-name'* AS *'mdx-expression'* ;

A calculated member is a dimension member that has been calculated from the member values in the input table. Only the definition of the member is stored; the value is calculated when a query is submitted. A named set is an alias for a specified MDX expression. Named sets are often used to make complex MDX queries easier to read and maintain.

The defined calculated members and named sets are available to any session that creates a query in the context of the SAS OLAP server and the schema defined in the METASRV statement of the PROC OLAP code that is used to create the global member or set.

DEFINE statements can apply to more than one cube, so the CUBE= option is not required to use this statement. The METASVR statement verifies that the cube definition exists in the metadata repository.

The DEFINE statement can be used alone as shown in this example, which defines two calculated members and one named set. The METASVR is the only other required statement. To define multiple sets or calculated members, separate option values with a comma.

```
proc olap;
metasvr olap_schema='Services Schema'
   repository='services'
   host='misdept.us.mar.com'
   port=9999
   protocol=bridge
   userid=jjones
   pw='my password'
   ;
defne member '[mddbcars].[Measures].[avg]' as
             '[Measures].[sales_sum]/[Measures].[sales_n]',
      member '[sales].[Measures].[stat1]' as
             '[Measures].[qty] +1',
```

```
      set '[campaign].[myset]' as
          '[campaign_dates].[All campaign_dates].children'
   ;
run;
```

The DEFINE statement can also be used with a PROC OLAP program that creates a cube or with a program that adds aggregations to or deletes aggregations from an existing cube. Cube builds, additions, and deletions occur before the DEFINE statement is processed, so the DEFINE statement is not processed if those statements fail.

```
proc olap data=olapsio.cars
   cube=mddbcars
   path='d:\services\'
   ;
metasvr olap_schema='Services Schema'
   repository='cars'
   host='misdept.us.mar.com'
   port=9999
   protocol=bridge
   userid=jjones
   pw='my password'
   ;
dimensoion date
   hierarchies=(date)
   sort_order=scending
   ;
hierarchy date
   LEVELS=(dte)
   ;
level dte
   ;
dimension cars
   hierarchies=(cars
   sort_order=ascending)
   ;
hierarchy cars
   levels=(car color)
   ;
dimension dealers
   hierarchies=(dealers)
   sort_order=ascending
   ;
hierarchy dealers
   levels=(dealer dest)
   ;
measure sales_sum
   column=sales
   stat=sum
   format=dollar15.2
   ;
measure sales_n
   column=sales
   stat=n
   format=12.0
   ;
```

```
define member '[mddbcars].[Measures].[avg]' as
   '[Measures].[sales_sum] / [Measures].[sales_n]'
   ;
run;
```

Required Arguments

MEMBER | SET
indicates whether you are creating a calculated member or a named set.

'member-or-set-name'
specifies the name of the member or set that you are creating. If you are creating a calculated member, then this value specifies a name for the member that will be calculated by the MDX expression. If you are creating a named set, then this value is the alias for the specified MDX expression.

AS *'mdx-expression'*
specifies the MDX expression.

UNDEFINE Statement

The UNDEFINE statement deletes from a SAS metadata repository one or more global calculated members or named sets.

UNDEFINE MEMBER | SET *'member-or-set-name'* ;

To delete multiple calculated members or named sets in a single UNDEFINE statement, use commas to separate instances of MEMBER | SET *'member-or-set-name'*.

For additional information on calculated members and named sets, see "DEFINE Statement" on page 102.

The following example shows how a single UNDEFINE statement can be used to delete from a metadata repository two calculated members and one named set.

```
proc olapl;
metasvr olpa_schema='Services Schema'
   repository='services'
   host='misdept.us.mar.com'
   port=9999
   protocol=bridge
   userid=jjones
   pw='my password'
   ;
undefine member '[carsCube].[Measures].[avg]',
   member '[sales].[Measures].[stat1]',
   set '[campaign].[myset]'
   ;
run;
```

Required Arguments

MEMBER | SET
indicates whether you are deleting a global calculated member or a named set.

'member-or-set-name'
specifies the name of the member or set that is to be deleted from the metadata repository. Cube names and dimension names are required for each *member-or-set-name*. Square brackets ([,])are optional inside the quotation marks of *member-or-set-name*, as shown in the preceding example.

USER_DEFINED_TRANSLATIONS Statement

The USER_DEFINED_TRANSLATIONS statement is required to use the Multiple Language Support capabilities of the SAS OLAP Server. This statement specifies the locales that are associated with the data sets that you specify in the DIMENSION statement.

USER_DEFINED_TRANSLATIONS *locale <locale2 ...localeN>* ;

Note: Alternative statement names are UDT and USER_DEFINED_TRANSLATION. △

PROC OLAP uses the UDT statement information, along with DIMENSION statement options, to read your alternate locale data sets and create locale-specific metadata for use at query time. Query results are returned in the language of the requested locale. The Multiple Language Support feature is available only for cubes that are loaded from a star schema. The alternate locale data set names consist of a prefix, which indicates the member, and a suffix, which indicates the language. The DEFINE statement supplies the suffix. The DIMTABLEMEMPREF= option in the DIMENSION statement specifies the member prefix. For example, if the member prefix is **dealdim_** and the suffix is **pl_PL**, then PROC OLAP looks for a data set named **dealdim_pl_PL.sas7bdat** in the library that is specified by the DIMTABLELIBREF= option.

The following sample code looks for these dimension data sets in the **mylib** library. The default locale is the first locale specified in the UDT statement. Additionally, the default locale does not use the suffix that is defined by the UDT statement. In this example, Polish is the default locale, so the suffix is not used.

```
dimension date
   hierarchies=(date)
   sort_order=ascending
   dimtablelibref=mylib
   dimtablemempref=ctimedim_
   factkey=dte
   dimkey=dte
   ;
hierarchy date levels=(dte)
   ;
level dte
   ;
dimension cars
   hierarchies=(cars)
   sort_order=ascending
```

```
   dimtablelibref=mylib
   dimtablemempref=cardim_
   factkey=carkey
   dimkey=carkey
   ;
dimension cars
   levels=(car color)
   ;
dimension dealers
   hierarchies=(dealers)
   sort_order=ascendng
   dimtablelibref=mylib
   dimtablemempref=dealdim_
   factkey=dealerkey
   dimkey=dealerkey
   ;
hierarchy dealers
   levels=(dealer dest)
   ;
user_defined_translations
   pl_PL   /* Polish as used in Poland */
   en_US   /* English as used in the United States */
   ja_JP   /* Japanese as used in Japan */
   ;
```

Table A1.7 Locales and Associated Data Set Names

Locale	Dimension Data Sets		
English	ctimedim_en_US	cardim_en_US	dealdim_en_US
Japanese	ctimedim_ja_JP	cardim_ja_JP	dealdim_ja_JP
Polish	ctimedim_	cardim_	dealdim_

Required Argument

locale
specifies the locales that correspond to the data sets contained in the library that is specified by the DIMTABLELIBREF= option in the DIMENSION statement. Separate locales with a space.

SAS Servers and Character Encoding

If your server metadata contains characters other than those typically found in English, then you must be careful to start your server with an ENCODING= or LOCALE= system option that accommodates those characters. For example, a SAS server started with the default US English locale cannot read metadata that contains Japanese characters. SAS will fail to start and will log a message indicating a transcoding failure.

In general, different SAS jobs or servers can run different encodings (such as ASCII/EBCDIC or various Asian DBCS encodings) as long as the encoding that is used by the particular job or server can represent all the characters of the data being processed. In the context of server start up, this requires that you review the characters used in the

metadata describing your server (as indicated by the SERVER= objectserverparm) to ensure that SAS runs under an encoding that supports those characters.

Tables Used to Define Cubes

There are five types of tables that can be used to define a cube:

- detail tables
- fact tables and dimension tables (for cubes that are based on star schemas)
- aggregation tables
- drill-through tables

Detail Tables

A detail, or base, table is any table that is defined in a SAS metadata repository that contains the columns for the measures and levels of a cube. A detail table consists of unsummarized data that must include one column for each level and one numeric analysis column for each set of measures that will be generated.

Fact Tables and Dimension Tables

A star schema refers to a set of input tables that are defined in a SAS metadata repository. A set of tables includes a single fact table and one or more dimension tables. A fact table must contain one numeric analysis column for each set of measures that will be generated. For levels, a fact table will either contain the columns for the levels of a dimension or contain a key column that links the fact table with a dimension table that contains the columns for the levels of a dimension.

The following statements are also true for star schemas:

- A fact table can contain a dimension. When this occurs, all the level columns are contained in the fact table and no fact or dimension key is required.
- If the dimension levels are defined in a dimension table, then all the level columns for that dimension must be contained in the same dimension table.
- Both the dimension keys and the fact keys are single columns, not combinations of columns.
- The dimension key can also be a level in the dimension.

Aggregation Tables

Aggregation tables are fully summarized, external relational tables. All aggregation tables must contain a column for each measure in the cube where the statistic for the measure is one of the following: N, NMISS, SUM, MAX, MIN, or USS. Columns for derived measures cannot be stored on the aggregation table and are ignored if they exist. Derived measures are always computed at query time. (See the STAT= option in the MEASURE statement for more information about stored and derived statistics.)

An aggregation table can be used in two ways:

- As an NWAY data source for the cube. In this case, the table must contain a column for every level in the cube and a column for every stored measure.
- As a subaggregation for the cube. In this case, the table must include a column for each level of the aggregation and a column for every stored measure.

Drill-Through Tables

Drill-through tables are views, data sets, or other data files maintained by the user that represent all of the relevant input data that is used to define a cube. This data is later accessed when performing drill-through actions from a client. PROC OLAP checks that the drill-through table has the correct level and measure names but never looks at the actual data contents of the table. The name of the drill-through table is stored in the cube's metadata in place of a detail or fact table name. (The detail or fact table

name is still referred to when an action is performed on a cube.) The drill-through table name can be set either when the cube is first created or during an update.

Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source.

Naming Guidelines for SAS OLAP Server

For SAS OLAP Server, names

- can be up to 32 characters in length.
- can contain mixed-case letters. SAS stores and writes the variable name in the same case that is used in the first reference to the variable. However, when SAS processes a variable name, SAS internally converts it to uppercase. You cannot, therefore, use the same variable name with a different combination of uppercase and lowercase letters to represent different variables. For example, cat, Cat, and CAT all represent the same variable.
- can contain characters other than Latin alphabet letters, numerals, and underscores, including embedded blanks. An exception to this is the dot character (.), which is an invalid character. You can do this by setting the SAS system option VALIDVARNAME=ANY. When this is set, PROC OLAP interprets the name as a *SAS name literal*, which is a token that is expressed as a string within quotation marks, followed by the letter *n*. Here are some examples:

```
dimesion 'Product@Work Dimension'n
   hierarchies=('Product@Work Hierarchy'n)
   ;
hierarchy "Product@Work Hierarchy"n
   levels=(prodtype product)
   ;
```

The PROC saves the case of the FIRST LOGICAL ENCOUNTER of an object name in the PROC script and saves it as the final stored name in the cube metadata. Therefore, the logical order of statement processing is as follows:

DIMENSION
HIERARCHY
LEVEL
AGGREGATION
MEASURE
PROPERTY.

Every DIMENSION statement is processed before any HIERARCHY statement and every HIERARCHY statement is processed before any LEVEL statement. This rule affects hierarchy and level names only. The stored hierarchy names are copied from the DIMENSION statement and the stored level names are copied from the first HIERARCHY statement listed in the PROC that uses this level. Here is an example:

```
hierarchy Time
   levels=(Year Month Day)
   ;
dimension Time
   hierarchies=(Time)
   ;
level year
```

```
type=year
;
```

In this example the hierarchy name is stored in the cube as "Time" rather than "TimE". The level name for year is stored as "Year" rather than "YEAR". The AGGREGATION statement does not affect the case of the level names that are stored in the cube metadata.

Note: For further information about the VALIDVARNAME= system option, see "VALIDVARNAME=System Option" and "Names in the SAS Language" in *SAS Language Reference: Dictionary*. △

Loading Cubes

Loading Cubes from a Detail Table

The following table lists the PROC OLAP statements and options that you use to load a cube from a detail table. The detail table has a column for each level and at least one numeric analysis column from which one or more measures can be generated.

Table A1.8 Statements and Options Used for Loading Cubes from a Detail Table

Use these statements	Use these options	
AGGREGATION	The AGGREGATION statement is optional unless you are creating additional aggregations, in which case, you must specify the names of the contiguous levels to be used to create the aggregation. Use the TABLE= option for cubes that are loaded from fully summarized tables.	
DIMENSION	HIERARCHIES=	Required
	DESC=	Optional
	CAPTION=	Optional
	TYPE=TIME	Required only for TIME dimensions
	SORT_ORDER=	Optional
HIERARCHY	LEVELS=	Required
	DESC=	Optional
	CAPTION=	Optional
LEVEL	DESC= CAPTION= TYPE= The LEVEL statement is optional unless you want to specify time periods for each level in a TIME dimension. If you specify a time period for one level, then you must specify a time period for all levels. To specify a time period, you use the TYPE= option.	
MEASURE	STAT=	Required
	COLUMN | ANALYSIS=	Required

Use these statements	Use these options	
	AGGR_COLUMN	Required if you use the AGGREGATION statement with the TABLE= option
	DESC=	Optional
	CAPTION=	Optional
	UNITS=	Optional
	FORMAT=	Optional
	DEFAULT=	Optional
METASVR	OLAP_SCHEMA=	Required
	REPOSITORY=	Optional
	HOST=	Optional
	PORT=	Optional
	PROTOCOL=	Optional
	USERID=	Optional
	PW=	Optional
PROC OLAP	DATA=	Required
	CUBE=	Required
	PATH=	Required
	DESC=	Optional
	NO_NWAY	Optional

Loading Cubes from a Star Schema

The following table lists the PROC OLAP statements and options that you use to load a cube from a star schema. A star schema is a set of input tables that are defined in a repository. The set of tables includes a single fact table and one or more dimension tables. The fact table must contain at least one numeric analysis column for each set of measures that will be generated.

Table A1.9 Statements and Options Used to Load Cubes from a Star Schema

Use these statements	Use these options	
PROC OLAP	FACT=	Required
	CUBE=	Required
	PATH=	Required
	DESC=	Optional
	NO_NWAY	Optional
METASVR	OLAP_SCHEMA=	Required
	REPOSITORY=	Optional
	HOST=	Optional

Use these statements	Use these options	
	PORT=	Optional
	PROTOCOL=	Optional
	USERID=	Optional
	PW=	Optional
DIMENSION	HIERARCHIES=	Required
	DESC=	Optional
	CAPTION=	Optional
	TYPE=TIME	Required only for TIME dimensions
	TYPE=GEO	Required only for GEO (ESRI Map) dimensions
	SORT_ORDER=	Optional
	DIMTBL=	Required for cubes that support one locale. If the cube will contain multiple national languages, replace this option with DIMTABLELIBREF= and DIMTABLEMEMPREF=.
	DIMKEY=	Required
	FACTKEY=	Required
	DIMTABLELIBREF=	Required if you are building a cube that will contain multiple national languages. Replaces DIMTBL=.
	DIMTABLEMEMPREF=	Required if you are building a cube that will contain multiple national languages. Replaces DIMTBL=.
LEVEL		The LEVEL statement is optional unless you want to specify time periods for each level in a TIME dimension. If you specify a time period for one level, then you must specify a time period for all levels. To specify a time period, you use the TYPE= option.
HIERARCHY	LEVELS=	Required
	DESC=	Optional
	CAPTION=	Optional
MEASURE	STAT=	Required
	COLUMN | ANALYSIS=	Required
	AGGR_COLUMN=	Required if you use the AGGREGATION statement with the TABLE= option

Use these statements	Use these options	
	DESC=	Optional
	CAPTION=	Optional
	UNITS=	Optional
	FORMAT=	Optional
	DEFAULT=	Optional
AGGREGATION		The AGGREGATION statement is optional unless you are creating additional aggregations, in which case, you must specify the names of the contiguous levels to be used to create the aggregation. Use the TABLE= option for cubes that contain aggregated data from tables other than the input data source.

Loading Cubes Using Summarized Data

The following table lists the PROC OLAP statements and options that you use to load cubes from a fully summarized data source (a crossing of all dimensions also known as an NWAY):

Table A1.10 Statements and Options Used to Load Cubes from Fully Summarized Data

Use these statements	Use these options	
PROC OLAP	CUBE=	Required
	PATH=	Required
	DESC=	Optional
METASVR	OLAP_SCHEMA=	Required
	REPOSITORY=	Optional
	HOST=	Optional
	PORT=	Optional
	PROTOCOL=	Optional
	USERID=	Optional
	PW=	Optional
DIMENSION	HIERARCHIES=	Required
	DESC=	Optional
	CAPTION=	Optional
	TYPE=TIME	Required only for TIME dimensions
	SORT_ORDER=	Optional

Use these statements	Use these options	
LEVEL		The LEVEL statement is optional unless you want to specify time periods for each level in a TIME dimension. If you specify a time period for one level, then you must specify a time period for all levels. To specify a time period, you use the TYPE= option.
HIERARCHY	LEVELS=	Required
	DESC=	Optional
	CAPTION=	Optional
MEASURE	STAT=	Required
	COLUMN | ANALYSIS=	Required
	AGGR_COLUMN=	Required
	DESC=	Optional
	CAPTION=	Optional
	UNITS=	Optional
	FORMAT=	Optional
	DEFAULT=	Optional
AGGREGATION	names of the contiguous levels to be used to create the aggregation	Required (additional AGGREGATION statements without the TABLE= option can be used to create aggregations other than the automatically defined NWAY)
	TABLE=	Required

Maintaining Cubes

Building a Cube from an Existing Definition

It is possible to have cube definitions in the SAS metadata repository that do not have associated physical cubes. For example, you can use the DELETE_PHYSICAL= option in the PROC OLAP statement to delete a cube but leave its definition intact. You can also use SAS OLAP Cube Studio to save only the definition of a new cube.

The following table lists the PROC OLAP statements and options that you use to build a cube from an existing metadata definition:

Table A1.11 Statements and Options Used to Build a Cube from an Existing Definition

Use these statements	Use these options
PROC OLAP	CUBE=
METASVR	OLAP_SCHEMA=

Adding Aggregations to an Existing Cube

The following table lists the PROC OLAP statements and options that you use to add aggregations to an existing cube.

Table A1.12 Statements and Options Used to Add Aggregations to an Existing Cube

Use these statements	Use these options	
PROC OLAP	CUBE=	Required
METASVR	OLAP_SCHEMA=	Required
AGGREGATION	Names of the contiguous levels to be used to create the aggregation	Required
	NAME=	Optional
	DATAPATH=	Optional
	INDEXPATH=	Optional
	COMPRESS | NOCOMPRESS	Optional
	INDEX | NOINDEX	Optional
	PARTSIZE=	Optional
	SEGSIZE=	Optional

Note: You can add and delete aggregations in the same PROC OLAP script. △

Note: You cannot add aggregations to a cube that contains aggregated data from a source other than the input data source. △

Deleting Aggregations from an Existing Cube

The following table lists the PROC OLAP statements and options that you use to delete aggregations from an existing cube.

Table A1.13 Statements and Options Used to Drop Aggregations from an Existing Cube

Use these statements	Use these options
DROP_AGGREGATION	Specify one or more level names that correspond to the aggregations that you want to remove, or use the aggregation name to specify the aggregation that you want to remove.
METASVR	OLAP_SCHEMA=
PROC OLAP	CUBE=

Note: You can add and delete aggregations in the same PROC OLAP script. △

Note: You cannot delete aggregations from a cube that contains aggregated data from a source other than the input data source. △

Deleting Cubes

The following table lists the PROC OLAP statements and options that you use to delete existing cubes.

If you use the DELETE option, then both the physical cube and its definition, which is stored in the metadata server, are deleted.

Table A1.14 Statements and Options Used to Delete a Cube and Its Metadata

Use these statements	Use these options
METASVR	OLAP SCHEMA=
PROC OLAP	CUBE=
	DELETE

If you use the DELETE_PHYSICAL option, then only the physical cube is deleted; the definition remains intact.

Table A1.15 Statements and Options Used to Delete a Cube but Retain Its Metadata

Use these statements	Use these options
METASVR	OLAP_SCHEMA=
PROC OLAP	CUBE=
	DELETE_PHYSICAL

Specialized Options for PROC OLAP

Options for Managing Ragged Hierarchies

If a hierarchy is balanced, then all of its branches descend to the same level, and each member has a parent level that is positioned immediately above it. However, hierarchies are not always balanced and sometimes they contain missing hierarchy members. To manage missing hierarchy members, you can use these four options, which were created specifically for ragged hierarchies:

Table A1.16 Options That Can Be Set to Manage Missing Hierarchy Members in Ragged Hierarchies

These options	Are available in these statements
EMPTY_CHAR=	PROC OLAP and HIERARCHY
EMPTY_NUM=	PROC OLAP and HIERARCHY

These options	Are available in these statements
EMPTY=	LEVEL
IGNORE_EMPTY	HIERARCHY and LEVEL

Options Used for Performance

When you create a cube, you can set some options that can be used to optimize cube creation and query performance. If you set the options in the PROC OLAP statement, then the settings are applied to all aggregations in the cube. If you set the options in the AGGREGATION statement, then the options apply to that specific aggregation. Options set for individual aggregations override any options set in the PROC OLAP statement.

The options are

- COMPACT_NWAY
- COMPRESS | NOCOMPRESS
- CONCURRENT=
- DATAPATH=
- INDEXPATH=
- INDEXSORTSIZE=
- MAXTHREADS=
- NOINDEX | INDEX
- PARTSIZE=
- SEGSIZE= .

Note: INDEXSORTSIZE=, MAXTHREADS=, and CONCURRENT= are available only on the PROC OLAP statement. △

For an explanation of these options, see the PROC OLAP statement and the AGGREGATION statement.

APPENDIX

2

SAS OLAP Cube Studio Messages

Cube Designer Error Messages

Table A2.1 Cube Designer Error Messages

Error Message	Comment
An aggregation cannot be named Default. Please enter another name.	This message is received on the Generated or User-Defined Aggregation pages. Rename the aggregation to a valid SAS name.
The map information is incorrect. To correct the information, select Yes and then display the Specify Map page for the geographic dimension. Select No to remove all the map information.	This message is received on the Dimension page when specifying GIS map information for a dimension.
The GEO dimension was changed to a STANDARD dimension and all map information has been removed.	This message is received on the Dimension page when specifying GIS map information for a dimension.
At least one dimension must be defined.	This message is received on the Dimension page. You must define a dimension for the cube.
A problem was encountered reading the map information.	This message is received on the Dimension page when specifying GIS map information for a dimension.
The <table name> table is used in a dimension but is not selected. Select OK to remove the dimension and aggregations created from this table or Cancel so you can select the table.	This message is received on the Dimension Tables page when removing the table associated with existing dimensions.
The <table name> tables are used in a dimension but are not selected. Select OK to remove the dimension and aggregations created from these tables or select Cancel so you can select the tables.	This message is received on the Dimension Tables page when removing more than one table.

Error Message	Comment
One or more tables must be selected as Dimension tables.	This message is received on the Dimension Tables page when a dimension table is not selected.
The user-defined aggregation already exists.	This message is received on the Generated Aggregations page when the selected levels match the levels in another aggregation.
The selected user-defined aggregation already exists.	This message is received on the Stored Aggregations page when the selected levels match the levels in another aggregation.
<level name> level does not follow drill-path order.	This message is received on either the Generated or User-Defined Aggregations page. The selected levels must match the drill order of at least one hierarchy.
Member Property name already exists for this level and hierarchy	This message is received on the Define a Member Property page. Click **OK** on the message and the duplicate member property name is checked. You must change the name for this member property to a unique name.
The aggregation name already exists.	This message is received on either the Generated or User-Defined Aggregations page. You must enter a different aggregation name.
Changes to metadata definitions detected. The library definition for the Drill-Through table is missing.	The dimension table metadata is checked on various functions in SAS OLAP Cube Studio, including the Create cube, Synchronize cube, Properties, Edit Cube Structure, and Manual Tuning dialog boxes. It is also checked on the Finish page of the Cube Designer wizard. If you receive this message, open Data Integration Studio and replace the missing libref for the table. Note: You can also receive the message when creating code in SAS Data Integration Studio.
Changes to metadata definitions detected. The library definition for the Drill-Through table is missing. Processing will continue.	See SAS Intelligence Platform and specifically SAS Management Console documentation for further information about metadata definitions.
Changes to metadata definitions detected. The library definition for one of the input tables is missing. Cube metadata will be saved, but the cube cannot be created.	See SAS Intelligence Platform and specifically SAS Management Console documentation for further information about metadata definitions.
Changes to metadata definitions detected. The PROC OLAP code cannot be generated because the library definition for one of the input tables is missing.	See SAS Intelligence Platform and specifically SAS Management Console documentation for further information about metadata definitions.

Error Message	Comment
Problems with dimension key metadata definitions have been automatically resolved. Use the Dimension Designer to verify automatically selected dimension keys. Do you accept the current dimension keys?	There are several locations in SAS OLAP Cube Studio where star schema metadata is verified, including the Create, Manual Tuning, and Save PROC OLAP code dialog boxes and the Synchronize levels function. The error is possibly with the dimension keys. You need to edit the cube structure and make sure automatically selected keys are acceptable. Note: You can also receive the message when generating code in SAS Data Integration Studio.
Problems with dimension key metadata definitions for the <dimension name> dimension have been automatically resolved. Use the Dimension Designer to verify automatically selected dimension keys.	See SAS Intelligence Platform and specifically SAS Management Console documentation for further information about metadata definitions.
Changes to cube metadata definitions detected. Edit the cube structure for automatic resolutions to the detected changes. Selecting Cancel at any time will not save metadata changes.	This message is received when there are changes to the input columns for levels and measures, and when aggregate columns for measures have been removed from the metadata. You will receive this message any time you receive other metadata messages. In addition, you will only be able to use the Edit Cube Structure function. You should open SAS Data Integration Studio to create new metadata for the missing items. You can then return to SAS OLAP Cube Studio and use the Edit Cube Structure function to repair the cube metadata.
Changes to metadata definitions detected. The <dimension name> star schema dimension is missing a key column definition. Modify the dimension in the Cube Designer to redefine the key column.	See SAS Intelligence Platform and specifically SAS Management Console documentation for further information about metadata definitions.
Changes to metadata definitions detected. The <dimension name> Dimension does not have all the tables for selected languages. Use the Cube Designer to verify selected languages or define language tables needed for this dimension.	See SAS Intelligence Platform and specifically SAS Management Console documentation for further information about metadata definitions.
Incomplete cube metadata detected. No input table. The cube must be deleted.	This message is received in the workspace when accessing various SAS OLAP Cube Studio Navigation Tree selections, including **`Create`**, **`Properties`**, and **`Edit Cube Structure`**.
Changes to metadata definitions detected. The PROC OLAP code cannot be saved because the library definition for one of the input tables is missing.	This message is specific to libref problems when you are selecting **`Save PROC OLAP code`** from the workspace or from the Finish page.
Changes to metadata definitions detected. The dimension key column for the <dimension name> dimension has changed. Modify the dimension to redefine the dimension key column.	This message is received if the keys cannot be repaired.

Error Message	Comment
The updates to the cube could not be saved. The deletion of the existing cube failed.	This message is received on the Finish page if the code submitted to the application server failed to delete the cube. The Cube Designer wizard cannot determine why the delete function failed. You should check the application server and try again.
The code to create the cube could not be submitted because the connection to a SAS Workspace Server failed.	You must be connected to a workspace server to create the physical cube. You should establish a connection to an application server and try to create the cube again.
The updates to the cube could not be saved. The existing cube must first be deleted. However, the connection to a SAS Workspace Server failed and the delete code could not be submitted.	This message is received when an existing cube needs to be deleted, but there is no application server connection. You should establish a connection to an application server and try to delete the cube again.
A cube type of HOLAP is specified, but no aggregation tables are identified. Pressing Finish will revert the cube type to MOLAP.	This message is received when you have clicked to the Finish page and a HOLAP-type cube does not have any aggregations. This is only a warning. You do not have to create stored aggregations.
Cube name must be unique.	You must enter a cube name that does not already exist within the selected OLAP schema.
Input type selection has changed. Clicking OK will remove the cube's current input-related definitions if they have been specified.	This message is received when the input type on the General page has changed.
The connection to a SAS Workspace Server failed. Unable to display the Browse dialog box.	This message is received when a connection to a workspace server has not been established and the **Browse** button is selected for either the Path or Work Path on the General page. Establish a connection to a workspace server and try again.
There are no OLAP schemas defined in the metadata.	This message is received when the **Next** button is selected on the General page and an OLAP schema has not been defined in the active metadata repository. Assign an OLAP schema to the active metadata repository and then define the cube.
An OLAP schema is not defined for this cube. Please select an OLAP Schema.	This message is received when an OLAP schema has not been selected on the General page. Select an OLAP schema from the drop-down list.
All stored aggregation definitions will be removed.	This message is received when the input type is changed to detail table or star schema, and the check box for **The cube will also use aggregated data from other tables** option has not been selected.

Error Message	Comment
You have changed the group name for a stored measure. Do you want to change the group name for all measures in that group? If you select no, the derived measures may be affected.	This message is received when you are defining stored measures for a cube and you change the Analysis Group name for a measure on the Assign Stored Measures page. Select **Yes** to change the group name for all measures in the analysis group.
A table must be selected as input to the cube.	This message is received on the Input page when you have not selected an input source for the cube you are creating. You must specify the data source that provides the input data for your cube.
Table selection has changed. Clicking OK will remove current cube definitions.	This message is received on the Input page when you reselect the input table. Selecting **OK** will reset all definitions in the cube.
The <statistic name> statistic is already used with the <group name> analysis group.	This message is received when you are defining stored measures for a cube and you assign a statistic that is already used on the Assign Stored Measures page.
The <measure name> derived measure is not possible.	This message is received when you are selecting measures for a cube. If stored aggregations exist for the cube, a derived measure cannot be created unless all required stored measures have already been created.
All aggregation column names must be the same on all tables. Select Back to select the correct tables, or don't select any tables.	This message is received on either the Select Measures page or the Generated Aggregations page. A table must have matching column names with other tables for the cube. You can modify the table in SAS Data Integration Studio and add the correct columns.
Measure name already exists for this cube.	The measure name already exists. You must specify a different measure name.
A measure name must be specified.	You must specify a name for the measure.
Setting the format to blank will set the format to the default format. There is no default format for this measure.	This message is received on the Measure Details page for HOLAP and MOLAP measures. If a format is not entered or selected and no default format exists, then this message is received.
Setting the format to blank will set the format to the default format. The default format for this measure is <format>.	This message is received on the Measure Details page for HOLAP and MOLAP measures. If a format is not entered or selected, then the default format will be used.
At least one measure must be defined for the cube.	This message is received on the Select Measures page. At least one measure must be selected for the cube.

Error Message	Comment
One or more derived measures statistics is not possible with the given groups. Do you want to continue?	This message is received when you are changing group names or statistics for a cube that has derived measures. The Assign Statistics Group page determines if any existing derived measures are invalid. You will receive this message if there are invalid derived measures.You should replace the incorrect statistics with valid statistics.
A group name must be assigned for each measure.	This message is received on the Assign Stored Measures page. An analysis group name must be assigned for each measure.
A statistic must be assigned for each measure.	This message is received on the Assign Stored Measures page. A statistic must be assigned for each measure.
No derived measures are possible with the currently defined stored measures. Check assigned analysis groups to make sure stored measures needed for a derived statistic have the same analysis group.	This message is received on the Select Derived Measures page. If you have stored measures already created for the cube, you cannot create derived measures for the cube. You can modify the analysis groups for these measures.
The limit of 1024 measures has been exceeded.	This message is received on the Select Measures page when you have exceeded the limit of 1024.
At least one non-Distinct Count (non-NUNIQUE) measure must be defined.	This message is received on the Select Measures page. You must select a measure from the list of available measures.
The SAS_SPATIAL_ID properties cannot be deleted here. To delete the property, edit the map information for the GEO dimension.	This message is received on the Member Property page. When ESRI map information is added to a cube, the property objects for the mapped cube levels are listed with the cube's member properties. These member properties are named SAS_SPATIAL_ID by default and cannot be deleted on this page.
You must specify an aggregate column for all stored measures. A stored measure is a measure that has SUM, N, NMISS, MIN, MAX or USS as the statistic.	This message is received on the Map Measures to Aggregated Columns page. When specifying stored measures, you must indicate an aggregate column.
You must select an aggregation table.	This message is received on the Specify a Stored User-Defined Aggregation page, which is accessed from the Stored Aggregations page. You must select an aggregation table when adding a stored aggregation.
A hierarchy is required to define a member property.	This message is received on the Define a Member Property page. You must have a hierarchy in the **`Selected Hierarchies`** list.
You must select a level to define a member property.	This message is received on the Define a Member Property page. You must have a level selected for the member property.

Error Message	Comment
You must select levels to define a user aggregation.	This message is received on the Manual Tuning dialog box. You must have one or more levels listed in the **Selected** list.
A name is required to define a member property.	This message is received on the Define a Member Property page. You must enter a name for the member property.
The aggregation name is required.	This message is received on the Generated Aggregations page when adding an aggregation. When specifying an aggregation, you must enter a name.
You must select a property for the level.	This message is received on the Member Property page. A column representing a member property hasn't been assigned to the level. Select a property for the level.
All stored measure names must be contained within the aggregation tables. Select the correct aggregation tables or don't select any tables to continue.	This message is received on the Measure Map page and the Generated Aggregations page. If the input to the cube is fully summarized, then the names of columns in the input table must match the names of the columns in a selected aggregation table. If not, then a different aggregation table needs to be selected.
Ragged hierarchy options for the <dimension name> Dimension must be corrected in Cube Designer.	This message is received when there is a problem with the current definition for ragged hierarchies. You must define the correct ragged hierarchy options for the cube. Click the **Advanced** button on the General page of the Cube Designer wizard. Select the **Ragged Hierarchies** tab and define the ragged hierarchies.
The specified cube name is not unique within the OLAP schema. Please enter a new name and select OK to complete the Cube Designer. Or select Cancel to return to the Cube Designer Finish page.	This message is received on the Cube Designer Finish page. While creating a cube, a different cube with the same name was saved to the application server. You should return to the General page and enter a new cube name.
A valid path must be entered to save the PROC OLAP code.	This message is received on the Save PROC OLAP Code dialog box when saving PROC OLAP code to a text file. You must enter a valid file system path to save the PROC OLAP code to.
The value for the COMPACT_NWAY option must be 0 or 1.	This message is received on the Advanced page that is accessed from the General page in the Cube Designer. This message is set internally to either 0 or 1 and is rarely received.
The <hierarchy name> levels must go from the most general time period to the most specific. The <level name> level does not follow this rule. Please modify the <hierarchy name> hierarchy.	When defining the hierarchy for a time dimension, you must list the levels in a specific order, going from general to specific in nature.

Error Message	Comment
Cannot add new measures. The maximum number of measures for a cube has been reached.	This message is received on the Select Measures page if the limit of 1024 measures is exceeded. It applies only to MOLAP-type cubes.
The aggregation table for this aggregation was deleted. Please delete this aggregation or go back and add the table that this aggregation was defined from.	This message is received when you modify a stored aggregation and the associated aggregation table has been removed. In this case the stored aggregations for the cube should also be removed.
Aggregation tables have not been selected. This will prevent any user-defined stored aggregations. Select OK to continue or Cancel to select aggregation tables.	This message is received on the Aggregation Tables page. You have not selected an aggregation table for the cube.
The aggregation tables you selected do not contain any levels within your cube. This will prevent any user-defined stored aggregations. Select OK to continue or Cancel to select aggregation tables.	This message is received on the Aggregation Tables page. If there are not any columns in the table that has been selected, then it is impossible to map the levels to that table. You can select another table or modify the table columns in SAS Data Integration Studio.

Dimension Designer Error Messages

Table A2.2 Dimension Designer Error Messages

Error Message	Comment
Enter a name for the dimension that is unique to a level name.	This message is received at the end of the Dimension Designer wizard on the Hierarchy page. When a new dimension is created, it is validated by SAS OLAP Cube Studio. If the dimension name matches one of the level names you will receive this message. Change the name of the dimension to something other than one of the selected level names.
Dimension name already exists for this cube.	This message is received on the Dimension Designer General page. The dimension name given matches the name of an existing dimension. You must enter a different dimension name.
Only one GEO dimension allowed for a cube.	This message is received on the Dimension Designer General page. There can be only one GEO-type dimension for a cube. Change the dimension type to STANDARD or TIME.

Error Message	Comment
Hierarchy name already exists for this dimension.	This message is received on the Dimension Designer Define a Hierarchy page. The hierarchy name has already been created for this dimension. You must enter a different hierarchy name.
Hierarchy name already exists for this cube.	The hierarchy name has already been created for this cube. You must enter a different hierarchy name.
Only one TIME dimension allowed for a cube.	This message is received on the Dimension Designer General page. You can only have one TIME dimension for a cube. Change the dimension type to STANDARD or GEO.
The hierarchy name cannot be the same as a level name.	This message is received on the Define a Hierarchy page. You cannot assign a name to a hierarchy that is the same as that for a level. You must enter a unique name.
The <level name> level is not defined in any hierarchy.	This message is received on the Hierarchy page. You must use all levels selected for the dimension in at least one hierarchy. If a level isn't used, you must either remove the level or assign it to a hierarchy.
A dimension name is required. Enter a new name that is not a level name.	This message is received on the Dimension Designer wizard. The dimension name must be unique and different from the names of the levels assigned to the dimension.
A dimension table must be selected.	This message is received on the Dimension Designer General page when defining star schema dimension tables. You must select a dimension table from the list box.
Dimension name is required.	This message is received on the Dimension Designer General page if a name is not given. You must enter a name for the dimension.
A dimension key is required.	This message is received on the Dimension Designer General page when defining star schema dimension tables. You must select a key from the list box.
A fact key is required.	This message is received on the Dimension Designer General page when defining star schema dimension tables. You must select a fact key from the list box.
Hierarchy name is required.	This message is received on the Define a Hierarchy page. You must enter a name for the hierarchy.
You must select levels.	This message is received on several locations, including the Dimension Designer Wizard, Manual Tuning dialog box, and the Defined Aggregations page. You must select one or more levels from the list of available levels.

Error Message	Comment
You must select levels for the hierarchy.	This message is received on the Define a Hierarchy page. You must select one or more levels from the list of available levels.
The <hierarchy name> hierarchy still contains different ragged hierarchy options. Please modify any hierarchy and specify the ragged options you would like for all hierarchies.	This message is received on the Define a Hierarchy page. You must enter the correct ragged hierarchy information.
The hierarchies contain different ragged hierarchy options. Please modify any hierarchy and specify the ragged options you would like for all hierarchies.	This message is received on the Define a Hierarchy page. You must enter the correct ragged hierarchy information.
Cannot add new dimensions. The maximum number of dimensions for a cube has been reached.	This message is received on the Cube Designer Dimensions page when the limit of 128 dimensions has been met.
Cannot add new hierarchies. The maximum number of hierarchies for a cube has been reached.	This message is received on the Hierarchy page when the limit of 128 hierarchies has been met.
Cannot add new levels. The maximum number of levels for a cube has been reached.	This message is received on the Cube Designer Dimensions page when the limit of 256 levels has been met. You will receive this message specifically on the following pages: □ the Cube Designer Dimensions page if you try to add another dimension □ the Dimension Designer Levels page if you try to create a new level
Cannot add new levels. The maximum number of levels for a dimension has been reached.	This message is received on the Dimension Designer Levels page if the number of selected levels exceeds 19.
Cannot add new member properties. The maximum number of member properties has been reached.	This message is received on the Member Property page. You cannot have more than 256 member properties per cube.
The WEEKS level type is not allowed if HALF_YEARS, QUARTERS or MONTHS are selected within the hierarchy.	This message is received on the Dimension Designer Levels page. You must select the correct combination of levels for a TIME-type dimension.

Specify Map Error Messages

Table A2.3 Specify Map Error Messages

Error Message	Comment
Some of the map information is missing. Please correct the problems.	This message is received on the Specify Map for Dimension page.
A column appropriate for the selected field ID must be selected.	This message is received on the Specify Map for Dimension page. You must select a Field ID column.
A map layer has been assigned but no field ID has been assigned.	This message is received on the Specify Map for Dimension page. You must assign a field ID.
There are no map field IDs associated with the selected map layer.	This message is received on the Specify Map for Dimension page and refers to an MdException problem with the metadata server. This message is uncommon.
There are no map layers associated with the selected map service.	This message is received on the Specify Map for Dimension page.
No layers have been assigned to the levels.	This message is received on the Specify Map for Dimension page. You must assign map layers to the cube levels.
A map server has been selected but no map service is selected.	This message is received on the Specify Map for Dimension page. You must select a map service.
There are no map services associated with the selected map server.	This message is received on the Specify Map for Dimension page.

Miscellaneous Error Messages

Table A2.4 Miscellaneous Error Messages

Cube Studio Category	Error Message	Comment
Cube Studio	A valid SAS name must be entered.	You must enter a valid SAS name. This message is received when SAS OLAP Cube Studio verifies the names for the cube, dimensions, hierarchies, measures, and member properties.
Model	The cube is missing the version property object.	The version property object must be identified.

Cube Studio Category	Error Message	Comment
Model	Cube metadata version <version number> is not supported.	This message is received when you attempt to create a model using a cube with a higher version than the model. This message appears before any Cube Designer pages appear.
Performance Option	(CONCURRENT) The number of aggregations to create in parallel must be an integer greater than or equal to 0.	This message is received on the Global Performance Options page if you enter a concurrent value below 0.
Performance Option	(INDEXSORTSIZE) The amount of memory (in megabytes) available for aggregation creation must be between 32 and 10239.	This message is received on the Global Performance Options page if you enter an index sort size between 32 and 10239.
Performance Option	(MAXTHREADS) The maximum number of threads used to create the aggregation index must be between 0 and 65536.	This message is received on the Global Performance Options page if you enter a maximum number of threads that is less than or equal to 0 or greater than or equal to 65536.
Performance Option	(PARTSIZE) The number of aggregations to create in parallel must be 16 or more.	This message is received on the **Generated Aggregations Performance Options** tabs for either Global or Aggregation options. The setting for **Partition size (in megabytes) of aggregation table partitions** must be 16 or more.
Performance Option	(SEGSIZE) The number of observations (in kilobytes) to include in the index component file segment must be 1 or more.	This message is received on the **Generated Aggregations Performance Options** tabs for either Global or Aggregation options. The setting for **Number of observations (in kilobytes) to include in the index component file segment** must be 1 or more.

APPENDIX

3

SAS OLAP Cube Studio Accessibility Features

SAS OLAP Cube Studio Accessibility Features

SAS OLAP Cube Studio includes the following accessibility and compatibility features that improve usability of the product for users with disabilities. These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended.

For further information on accessibility features in SAS, see *Accessibility Features in SAS under Windows* and *Default Key Settings for Interactive SAS Sessions under Windows* in SAS OnlineDoc.

Note: If you have questions or concerns about the accessibility of SAS products, send e-mail to **`accessibility@sas.com`**. △

Table A3.1 Window Manipulation Keys

Window Manipulation Key	Action
ALT+F4	Closes active applications on the desktop. ALT+F4 Closes SAS OLAP Cube Studio.
ALT+F6	Switches to the next window between modeless secondary windows and their primary window.
ALT+SPACEBAR	Opens the title bar icon and displays the program menu of the leftmost icon on the title bar.
ALT+TAB	Provides a pop-up menu of active applications, identified by their product graphics.
ALT+TAB+SHIFT	Reverses direction through the pop-up menu.
ALT+HYPHEN	Displays the shortcut menu for the active MDI window.
ALT+ESC	Navigates active applications on the desktop and then desktop shortcut bar items. Releasing the ESC key selects an application.

Window Manipulation Key	Action
ALT+ESC+SHIFT	Reverses the navigation direction of active applications on the desktop and the desktop shortcut bar. Navigates windows in reverse order.
ALT+ENTER	Displays the properties of a selected item while you are working in a window.
PRINT SCREEN	Copies a screen image to the clipboard.
ALT+ PRINT SCREEN	Copies an active window image to the Windows clipboard.

Table A3.2 Actions Performed with the ALT Key.

Action	Procedure
Move the application window	In SAS OLAP Cube Studio, select the ALT key and perform the following steps: **1** Choose **Move** from title bar menu. **2** Use the arrow keys to move the window. **3** Press ENTER to accept or ESC to cancel.
Size the application window	In SAS OLAP Cube Studio, select the ALT key and perform the following steps: **1** Choose **Size** from title bar menu. **2** Use an arrow key to choose which window border to move. **3** Resize the window with the appropriate arrow keys. **4** Press ENTER to accept or ESC to cancel.
Minimize/Maximize the application window	In SAS OLAP Cube Studio, perform the following steps: **1** Choose **Minimize** from the title bar menu. **2** Choose **Maximize** from the title bar menu.
Restore the application window	In SAS OLAP Cube Studio, select the ALT key and select **Restore** from title bar menu. The application is returned to its original state.

APPENDIX

4

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *Administrator for Enterprise Clients: User's Guide*
- *SAS Data Providers: ADO/OLE DB Cookbook*
- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Management Console: User's Guide*
- *SAS Intelligence Platform: System Administration Guide*
- *SAS Intelligence Platform: Data Administration Guide*
- *SAS OLAP Server: Concepts and Excerpts from "MDX Solutions with Microsoft SQL Server Analysis Services"*
- *SAS OLAP Server: MDX Guide*
- SAS Companion that is specific to your operating environment

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

aggregation
a summary of detail data that is stored with or referred to by a cube. Aggregations support rapid and efficient answers to business questions.

aggregation table
a table that contains pre-calculated totals. Aggregation tables can be referred to by cubes, reducing the amount of time that is required for building the cubes.

ARM (Application Response Measurement)
an application programming interface that was developed by an industry partnership and which is used to monitor the availability and performance of software applications. ARM monitors the application tasks that are important to a particular business.

base table
a table that contains detail data that is used for building cubes or aggregation tables.

calculated member
in a dimension, a member whose value is derived from the values of other members.

child
within a dimension hierarchy, a descendant in level n-1 of a member that is at level n. For example, if a Geography dimension includes the levels Country and City, then Bangkok would be a child of Thailand, and Hamburg would be a child of Germany.

cube
a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement. A cube is a directory structure, not a single file. A cube includes measures, and it can have numerous dimensions and levels of data.

custom repository
in the SAS Open Metadata Architecture, a metadata repository that must be dependent on a foundation repository or custom repository, thus allowing access to metadata definitions in the repository or repositories on which it depends. A custom repository is used to specify resources that are unique to a particular data collection. For example, a custom repository could define sources and targets that are unique to a particular data warehouse. The custom repository would access user definitions, group definitions, and most server metadata from the foundation repository. See also foundation repository, project repository.

data cleansing
the process of eliminating inaccuracies, irregularities, and discrepancies from data.

data scrubbing
another term for data cleansing. See data cleansing.

data sparsity
a characteristic of a multidimensional data source in which there is a relatively high proportion of empty cells (which indicate missing data values) to filled cells.

data warehouse
a collection of data that is extracted from one or more sources for the purpose of querying and analysis.

descendant
in a dimension hierarchy, a member that resides at a lower level in relation to other members in the hierarchy. For example, if a Geography dimension includes the levels Country, State, and City, then California and Los Angeles would be descendants of USA.

detail data
nonsummarized (or partially summarized) factual information that pertains to a single area of interest, such as sales figures, inventory data, or human-resource data.

dimension
a group of closely related hierarchies. Hierarchies within a dimension typically represent different groupings of information that pertains to a single concept. For example, a Time dimension might consist of two hierarchies: (1) Year, Month, Date, and (2) Year, Week, Day. See also hierarchy.

dimension table
in a star schema, a table that contains the data for one of the dimensions. The dimension table is connected to the star schema's fact table by a primary key. The dimension table contains fields for each level of each hierarchy that is included in the dimension.

drill down
in a view of an OLAP cube, to start at one level of a dimension hierarchy and to click through one or more lower levels until you reach the data that you are interested in.

drill up
in a view of an OLAP cube, to start at one level of a dimension hierarchy and to click through one or more higher levels until you reach the level of summarized data that you are interested in.

drill-through table
a view, data set, or other data file that contains data that is used to define a cube. Drill-through tables can be used by client applications to provide a view from processed data into the underlying data source.

fact
a single piece of factual information in a data table. For example, a fact can be an employee name, a customer's phone number, or a sales amount. It can also be a derived value such as the percentage by which total revenues increased or decreased from one year to the next.

fact table
the central table in a star schema. The fact table contains the individual facts that are being stored in the database as well as the keys that connect each particular fact to the appropriate value in each dimension.

foreign key
a column or combination of columns in one table that references the corresponding primary key in another table. A foreign key must have the same attributes as the primary key that it references.

foundation repository
in the SAS Open Metadata Architecture, a metadata repository that is used to specify metadata for global resources that can be shared by other repositories. For example, a foundation repository is used to store metadata that defines users and groups on the metadata server. Only one foundation repository should be defined on a metadata server. See also custom repository, project repository.

granularity
the relative level of detail that a data item represents. From the top of a dimension to the bottom, granularity increases. For example, in a Time dimension that consists of a Year-Month-Day hierarchy, Month is more granular than Year, and Day is more granular than Month.

hierarchy
an arrangement of members of a dimension into levels that are based on parent-child relationships. Members of a hierarchy are arranged from more general to more specific. For example, in a Time dimension, a hierarchy might consist of the members Year, Quarter, Month, and Day. In a Geography dimension, a hierarchy might consist of the members Country, State or Province, and City. More than one hierarchy can be defined for a dimension. Each hierarchy provides a navigational path that enables users to drill down to increasing levels of detail. See also member, level.

HOLAP (hybrid online analytical processing)
a type of OLAP in which relational OLAP (ROLAP) and multidimensional OLAP (MOLAP) are combined. In HOLAP, the source data is usually stored using a ROLAP strategy, and aggregations are stored using a MOLAP strategy. This combination usually results in the smallest amount of storage space. In HOLAP, aggregates can be pre-calculated and can be linked into a hybrid storage model.

informat
a pattern or set of instructions that SAS uses to determine how data values in an input file should be interpreted. SAS provides a set of standard informats and also enables you to define your own informats.

leaf member
the lowest-level member of a hierarchy. Leaf members do not have any child members.

level
an element of a dimension hierarchy. Levels describe the dimension from the highest (most summarized) level to the lowest (most detailed) level. For example, possible levels for a Geography dimension are Country, Region, State or Province, and City.

logical server
in the SAS Metadata Server, the second-level object in the metadata for SAS servers. A logical server specifies one or more of a particular type of server component, such as one or more SAS Workspace Servers.

MDDB (multidimensional database)
another term for cube. See cube.

MDX (multidimensional expressions) language
a standardized, high-level language that is used for querying multidimensional data sources. The MDX language is the multidimensional equivalent of SQL (Structured Query Language).

measure
a special dimension that contains summarized numeric data values that are analyzed. Total Sales and Average Revenue are examples of measures. For example, you might drill down within the Clothing hierarchy of the Product dimension to see the value of the Total Sales measure for the Shirts member.

member
a name that represents a particular data element within a dimension. For example, September 1996 might be a member of the Time dimension. A member can be either unique or non-unique. For example, 1997 and 1998 represent unique members in the Year level of a Time dimension. January represents non-unique members in the Month level, because there can be more than one January in the Time dimension if the Time dimension contains data for more than one year.

metadata profile
a definition of where a metadata server is located. The definition includes a host name, a port number, and a list of one or more metadata repositories. In addition, the metadata profile can contain a user's login information and instructions for connecting to the metadata server automatically.

metadata repository
a collection of related metadata objects, such as the metadata for a set of tables and columns that are maintained by an application. A SAS Metadata Repository is an example.

metadata server
a server that provides metadata management services to one or more client applications. A SAS Metadata Server is an example.

MOLAP (multidimensional online analytical processing)
a type of OLAP that stores aggregates in multidimensional database structures.

multi-threading
See threading.

navigate
to purposefully move from one view of the data in a table (or in some other data structure, such as a cube) to another. Drilling down and drilling up are two examples of navigation.

NWAY aggregation
the aggregation that has the minimum set of dimension levels that is required for answering any business question. The NWAY aggregation is the aggregation that has the finest granularity. See also granularity.

OLAP (online analytical processing)
a software technology that enables users to dynamically analyze data that is stored in multidimensional database (MDDB) tables.

OLAP schema
a group of cubes. A cube is assigned to an OLAP schema when it is created, and an OLAP schema is assigned to a SAS OLAP Server when the server is defined in the metadata. A SAS OLAP Server can access only the cubes that are in its assigned OLAP schema.

OLE DB for OLAP
an OLAP API that is used to link OLAP clients and servers by means of a multidimensional expressions (MDX) language. See also MDX (multidimensional expressions) language.

parallel I/O
a method of input and output that takes advantage of multiple CPUs and multiple controllers, with multiple disks per controller to read or write data in independent threads.

parallel processing
a method of processing that divides a large job into several smaller jobs that can be executed in parallel on multiple CPUs.

parent
within a dimension hierarchy, the ancestor in level n of a member in level n-1. For example, if a Geography dimension includes the levels Country and City, then Thailand would be the parent of Bangkok, and Germany would be the parent of Hamburg. The parent value is usually a consolidation of all of its children's values.

primary key
a column or combination of columns that uniquely identifies a row in a table.

project repository
a repository that must be dependent on a foundation repository or custom repository that will be managed by the Change Management Facility. A project repository is used to isolate changes from a foundation repository or from a custom repository. The project repository enables metadata programmers to check out metadata from a foundation repository or custom repository so that the metadata can be modified and tested in a separate area. Project repositories provide a development/testing environment for customers who want to implement a formal change management scheme. See also custom repository, foundation repository.

ROLAP (relational online analytical processing)
a type of OLAP in which the multidimensional data is stored in a relational database.

roll up
to summarize (or apply some other type of calculation or formula to) data values at one level of a dimension hierarchy in order to derive values for a parent level. For example, sales figures for January can be rolled up to Quarter1, and employee data for one department can be rolled up to the division level.

SAS application server
a server that provides SAS services to a client. In the SAS Open Metadata Architecture, the metadata for a SAS application server specifies one or more server components that provide SAS services to a client.

SAS ARM interface
an interface that can be used to monitor the performance of SAS applications. In the SAS ARM interface, the ARM API is implemented as an ARM agent. In addition, SAS supplies ARM macros, which generate calls to the ARM API function calls, and ARM system options, which enable you to manage the ARM environment and to log internal SAS processing transactions. See also ARM (Application Response Measurement).

SAS format
a pattern or set of instructions that SAS uses to determine how the values of a variable (or column) should be written or displayed. SAS provides a set of standard formats and also enables you to define your own formats.

SAS informat
a pattern or set of instructions that SAS uses to determine how data values in an input file should be interpreted. SAS provides a set of standard informats and also enables you to define your own informats.

SAS Management Console
a Java application that provides a single user interface for performing SAS administrative tasks.

SAS Metadata Repository
one or more files that store metadata about application elements. Users connect to a SAS Metadata Server and use the SAS Open Metadata Interface to read metadata from or write metadata to one or more SAS Metadata Repositories. The metadata types in a SAS Metadata Repository are defined by the SAS Metadata Model.

SAS Metadata Repository
a repository that is used by the SAS Metadata Server to store and retrieve metadata. See also SAS Metadata Server.

SAS Metadata Server
a multi-user server that enables users to read metadata from or write metadata to one or more SAS Metadata Repositories. The SAS Metadata Server uses the Integrated Object Model (IOM), which is provided with SAS Integration Technologies, to communicate with clients and with other servers.

SAS name
a name that is assigned to items such as SAS variables and SAS data sets. The first character must be a letter or an underscore. Subsequent characters can be letters, numbers, or underscores. Blanks and special characters (except the underscore) are not allowed. The maximum length of a SAS name depends on the language element that it is assigned to. Many SAS names, such as names of DATA step variables and array names, can be 32 characters long. Others, such as librefs and filerefs, have a maximum length of 8 characters.

SAS OLAP Cube Studio
a Java interface for defining and building OLAP cubes in SAS System 9 or later. Its main feature is the Cube Designer wizard, which guides you through the process of registering and creating cubes.

SAS OLAP Server
a SAS server that provides access to multidimensional data. The data is queried using the multidimensional expressions (MDX) language.

SAS Open Metadata Architecture
a general-purpose metadata management facility that provides metadata services to SAS applications. The SAS Open Metadata Architecture enables applications to exchange metadata, which makes it easier for these applications to work together.

schema
a map or model of the overall data structure of a database. An OLAP schema specifies which group of cubes an OLAP server can access.

scrubbing
another term for data cleansing. See data cleansing.

slice
a subset of data from a cube, where the data in the slice pertains to one or more members of one or more dimensions. For example, from a cube that contains data about customer feedback, one slice might pertain to feedback on one particular product (one member of the Product dimension). Another slice might pertain to feedback on that product from customers residing in particular geographic areas who submitted their feedback during a certain time period (one member of the Product dimension, multiple members of the Geography dimension, one or more members of the Time dimension).

SMP (symmetric multiprocessing)
a hardware and software architecture that can improve the speed of I/O and processing. An SMP machine has multiple CPUs and a thread-enabled operating system. An SMP machine is usually configured with multiple controllers and with multiple disk drives per controller.

sparsity
See data sparsity.

SPD (Scalable Performance Data) Engine
a SAS engine that is able to deliver data to applications rapidly because it organizes the data into a streamlined file format. The SPD Engine divides a problem (such as a WHERE clause) into smaller problems that can be processed in parallel. See also parallel processing.

SQL (Structured Query Language)
a standardized, high-level query language that is used in relational database management systems to create and manipulate database management system objects.

star schema
tables in a database in which a single fact table is connected to multiple dimension tables. This is visually represented in a star pattern. SAS OLAP cubes can be created from a star schema.

stored statistics
statistics that are stored in a cube. Stored statistics can be used to derive higher-level statistics. Examples include sum, minimum, and maximum.

thread
a single path of execution of a process in a single CPU, or a basic unit of program execution in a thread-enabled operating system. In an SMP environment, which uses multiple CPUs, multiple threads can be spawned and processed simultaneously. Regardless of whether there is one CPU or many, each thread is an independent flow of control that is scheduled by the operating system. See also SMP (symmetric multiprocessing), thread-enabled operating system, threading.

thread-enabled operating system
an operating system that can coordinate symmetric access by multiple CPUs to a shared main memory space. This coordinated access enables threads from the same process to share data very efficiently.

threading
a high-performance method of data I/O or data processing in which the I/O or processing is divided into multiple threads that are executed in parallel. In the boss-worker model of threading, the same code for the I/O or calculation process is executed simultaneously in separate threads on multiple CPUs. In the pipeline model, a process is divided into steps, which are then executed simultaneously in separate threads on multiple CPUs. See also parallel I/O, parallel processing, SMP (symmetric multiprocessing).

Time dimension
a dimension that divides time into levels such as Year, Quarter, Month, and Day.

tuple
a data object that contains two or more components. In OLAP, a tuple is a slice of data from a cube. It is a selection of members (or cells) across dimensions in a cube. It can also be viewed as a cross-section of member data in a cube. For example, ([time].[all time].[2003], [geography].[all geography].[u.s.a.], [measures].[actualsum]) is a tuple that contains data from the Time, Geography, and Measures dimensions.

wizard
an interactive utility program that consists of a series of dialog boxes, windows, or pages. Users supply information in each dialog box, window, or page, and the wizard uses that information to perform a task.

Index

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Your Turn

If you have comments or suggestions about *SAS 9.1.3 OLAP Server: User's Guide, Second Edition*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**