ABSTRACT
The amount of data that is captured electronically is increasing at an exponential rate. While advances in scalable storage solutions allow companies to house this data, those tasked with analyzing and summarizing it now find themselves faced with the following questions:

• How do we turn raw data into useful information?
• How do we surface this information in a meaningful way?
• How do we provide access to all facets of the information that people may need to see?

To tackle these and other dilemmas, more and more organizations are turning to OLAP Cubes. OLAP, or Online Analytical Processing, provides a multi-dimensional view of aggregated data. It provides quick and easy access to facts, figures and statistics and gives the user the flexibility to change the view of the data to hone in on the aspects that are most pertinent to them. With the latest incarnation of SAS Enterprise Guide, SAS Programmers can also take advantage of the benefits of storing data in OLAP cubes.

This paper will discuss:

• The benefits of using OLAP Cubes
• Navigating OLAP Cubes with Enterprise Guide
• Querying Cubes with the built-in Cube Viewer
• Creating SAS datasets from cube slices
• Using MDX in PROC SQL to query an OLAP cube in your code

Enterprise Guide version 4.1 will be used.

Keywords: Enterprise Guide, OLAP, SQL, MDX

WHAT IS OLAP?
Many user-applications require immediate results based on huge volumes of data. Most users do NOT want to submit a job that churns away for hours before providing their results. With that in mind, if an application can be sourced from summarized data, then OLAP can help. OLAP, or Online Analytical Processing, allows a user to access and analyze summarized data (an OLAP cube) including the ability to quickly switch to different views of the aggregated business data (Slice and Dice).

OLAP TERMINOLOGY
Let’s start by tackling some basic OLAP terminology. In the cube above, the colors represent different groupings of data into logical categories, called dimensions. Dimensions might include Product, Geography, Job Roles, Time, etc. The order in which you get information from the summary down to the
detailed data is specified by a **hierarchy**. For example, the Geography dimension might be ordered as continent→country→province→city, with continent being the highest level of aggregation and city being the most detailed. The data values that are summarized and used for analysis are **measures**. A **slice** is a subset of the data that is available in the cube. The different groups of data that are summarized for easy and fast access are referred to as **aggregations**.

Cubes can be created using a single source of underlying data or can be created by using a series of tables connected by various keys. When creating a cube from a single table, you source the data from a **Detail Table** which contains the measures and levels (for defining the hierarchies). When using a series of tables, the underlying data structure is a **Star Schema**, which uses a Fact table along with a set of Dimension tables to define the cube.

**WHAT IS THE BENEFIT FOR YOU – THE SAS PROGRAMMER?**
We know that OLAP cubes provide business users with quick and easy access to data but what are the benefits for us SAS Programmers?

- Data in an OLAP cube is already aggregated, so half the work is already done for you when you extract the data
- Querying OLAP cubes through **passthru to OLAP** (using SAS OLAP Server’s multithreaded query engine) is much faster than accessing raw (detail) data
- You can create the same reports, but the jobs run much quicker
- Measures that you would otherwise calculate in your SAS code are already available and defined within the Cube
- **Drill Down** (investigating underlying data) is possible
- You still have the flexibility to manipulate the extracted cube data using your own SAS code.

**CREATING AND VIEWING OLAP CUBES**
There is more than one way to create a SAS cube. If you prefer to use a GUI based tool, you can use the Olap Cube Wizard (available in SAS Olap Cube Studio and Data Integration Studio). If you are a coder at heart, you can create, delete and update a SAS cube using PROC OLAP code. For those of you who are somewhere in between, you can use the OLAP Cube Wizard to generate the PROC OLAP code and then tweak and customize it yourself.

**PROC OLAP**
This example uses PROC OLAP to create a cube based on a **single detail table**, the Shoes dataset in the SASHELP library. The cube was created using the **Program_1** SAS code included in the appendix. This code will register the metadata for your underlying Detail table; it will create the OLAP cube and register the cube details in the metadata server. Let’s take a quick look at the PROC OLAP syntax.

**Cube basics – name, source table(s), path, description**
```
PROC OLAP Data=&libname..Shoes   DrillThrough_Table=&libname..shoes
cube=&cube   Path="&cubepath"  Description="Shoes Cube"   ;
```

**Specify the metadata connection details**
```
METASVR repository=< >   olap_schema=< > server=< > port=< >;
```

**Define dimensions and Hierarchies**
```
DIMENSION < > hierarchies=(< > )  CAPTION= < >  SORT_ORDER= < > ;
HIERARCHY < >  ALL_MEMBER=' < > '   levels=( < > )  CAPTION='< > '  
```

**Define measures based on built in metrics (ie. sum, average, mean etc)**
```
MEASURE < >
STAT=SUM   COLUMN= <>
CAPTION= ' <> '  FORMAT= <> ;
```

**Specify pre-aggregations**
```
AGGREGATION < >   / NAME=' < > ' ;
```
**Define custom calculations/measures that you’d like to include**

`DEFINE MEMBER '<[>]' AS '<>' ;`

In addition to the PROC OLAP code, we’ve included some additional steps in this program. The PROC METALIB step will refresh the metadata for the underlying data table, in our case a copy of the SASHELP.SHOES dataset. It is important to refresh the metadata for the underlying table since your table structure could have changed to include additional fields or different formats. We’ve also set many of the metadata system options for the session, saving us from having to specify the metadata details in the OLAP procedure. This is something that you may want to consider doing if you’re building several OLAP cubes in the same program.

Presently, SAS does not support incremental Cube updates so the cube has to be reloaded with all data each time it is refreshed. If your process updates or refreshes existing cubes, you will need to add a PROC OLAP DELETE step to your code:

```plaintext
PROC OLAP cube=Cube Name <DELETE OPTION>;  
METASVR olap_schema="&Schema"  
    repository="&repository";  
Run;
```

There are two DELETE options:

- **DELETE** – will delete the physical cube and all metadata associated with the cube. Use this option when you are changing or adding dimensions, hierarchies, measures or descriptions.
- **DELETE_PHYSICAL** – will only delete the physical data stored in the cube. Use this option when you want to change or update or add data to in the cube without changing the structure.

Note: Incremental updates will be introduced in 9.2.

**OLAP CUBE VIEWER**

Once the OLAP cube has been built and access privileges have been set, you can use the OLAP Cube Viewer in Enterprise Guide to navigate and drill through the cube.

- To open the Olap Cube in Enterprise Guide, click **File ➔ Open ➔ Olap Cube**
- Dependent upon your setup, enter the following into the Open OLAP Window (place information specific to your setup where the *** appear):
  - Olap Server Name: ***************
  - Provider: SAS OLAP Data Provider 9.1 (from drop down list)
  - User Name: ******
  - Password: *******
- Click **Connect**
- Check the box next to your newly created Cube – click **Open**.

Here is what you’ll see.....

The OLAP Viewer automatically opens displaying the default view of your cube. In this example, we specified the Division Hierarchy as our default (see Program_1 code), so this is the first view we see. Notice that the view is also aggregated to the highest level. You’ll also notice that the OLAP Viewer includes a Graphical representation of your current view. As you drill to different dimensions and levels, the graph automatically changes to reflect the data in the current view. You can remove the graph from the view to maximize the area available for viewing the
data (Click MORE in your Cube Viewer Task Bar, click GRAPH).

Cube View Manager: Allows you to display various dimensions and hierarchies in the table and graph view, customize preferences for the current cube and create shortcuts and filtered views of the data.

Table View: Shows cube data in a table format and allows you to expand levels and drill down on the data.

Graph View: Automatically updates to reflect the data shown in the table view. You can customize the graph layouts and export to other Windows tools.

Cube Viewer Menu Bar: Shortcut buttons to toggle between views (back, forward), customize view (View designer), view the MDX behind the current view, add new measures, filter the data and turn graph and tree views on and off.

You can close the Cube View Manager to create more space by clicking the ‘X’ in the upper right-hand corner of the Cube View Manager window. Note that these options (Dimensions, Preferences and Shortcuts) are now hidden tabs along the left hand side of your cube viewer and can be viewed simply by clicking on them.

Let’s explore this cube.

- Drill Down on the Product Type – click the downward facing arrow on the ‘All Product Type’ column.
- Expand the list of Divisions – click the plus sign on the ‘All Division’ Row.
You'll notice that you now have a **Slicer** view above your table view. The **cube slicer** allows you to filter the various dimensions. In this view we are looking at ALL product types and ALL regions.

Let's drill back up to the aggregated view of all products. Right click on the Product Hierarchy cell (where the Hierarchy level name is in Italic).

- Select **Drill Up To → All Product_Type**

Change the displayed measure. Right click on the 'MeasureLevel.' and select 'Edit Measures'

- Select 'Average Sales' and 'Sum of Sales'
- Click OK.
• Take a closer look at Canada

• Right click anywhere on the Canada cell. Click 'Isolate Canada'

Move the Products dimension from the columns to the rows. Right click on any of the Product Column headings, click Move Product to rows.

• Drill down on the Region to see the subsidiaries. Click the downward facing arrow on the Canada column heading.

• What if we wanted to see a breakdown of products by region? Right click on the ALL Product Type Dimension level, select Drill Down to Product.

Notice that our Slicer now tells us that we are looking at All Product Types in the Product dimension but only Canada in the Division Dimension. When slicing and dicing Cubes, you can sometimes lose track of the path you’ve taken to get a certain view. The Slicer view helps you keep track of what subset of data you’re looking at.
There are other features that are available in the cube viewer that help when exploring your cube data. Among other things, the cube viewer allows you to filter the data based on constraints that you define, you can order the data, you can create bookmarks or shortcuts to certain views, and you can create your own custom measures that may not already exist in the cube definition. Once you are comfortable with slicing and dicing your data views, I would encourage you to become familiar with some of these additional features available.

**INCORPORATING OLAP INTO YOUR SAS PROGRAMS**

Now that the OLAP cube is created and we have a better idea of the data available in the cube, we want to be able to use the aggregated data to create some reports. We’ve heard that taking advantage of organized dimensions, hierarchies and pre-aggregated data makes querying an OLAP cube is a lot faster than querying raw data tables but how do we actually query a cube from our SAS code?

Using PROC SQL (specifically the SQL pass-through facility for OLAP), you can connect to your cube and query the data like you would a SAS dataset or Database table. The query language that you use is called MDX and while it uses the familiar SQL keywords (select, from, where), that’s where the similarities end. So how do you connect to a cube through your code? First, let’s look at the familiar syntax (see Program_2 in Appendix)

```sas
Connect to OLAP (<options>)
Create table MYDATA as select * from connection to OLAP
( select <MDX query here>)
Disconnect from OLAP;
Quit;
```

Before we tackle the MDX query, let’s take a quick look at the options needed to connect to the cube:

```sas
Connect to OLAP (host = Server Name
Port = OLAP Server Port Number
Protocol =Protocol used to connect to OLAP cube(Bridge or COM)
User = User ID
Pass = User Password
Repository =Metadata Repository
Olap_schema =The schema in which your cube is defined);
```

Many of these options will be provided to you by your Administrator. Building the MDX select code is a *little* trickier:
SELECT {axisSpecification} ON COLUMNS,
     {axisSpecification} ON ROWS
FROM cubeName
WHERE {slicerItems}

Since Enterprise Guide has a built-in cheat sheet within its OLAP Cube Viewer, you can get away with not knowing much (if any) MDX syntax. You can use the MDX Editor in the Cube Viewer to build and tweak your query code. Simply navigate to the desired Cube view and copy the MDX query code that is built for you.

Let's go back to our Olap Cube viewer and get our MDX Query code.

- To view the MDX code from the Main Menu Select OLAP ➤ View MDX or click the MDX Editor button in your Cube Viewer Window
- Copy the MDX Query. Highlight the text, right click and select Copy.
- Paste the MDX query into your PROC SQL Code. Place your cursor where you want to insert the code, right click and hit Paste.
- Submit the code.

The data pulled by the PROC SQL step is available to the rest of your SAS code as a SAS dataset. Note that since the data we extracted is already pre-aggregated, we don’t have to perform any additional summarizing on the data. A simple proc report is run on the data to output it in the appropriate format.

In addition to querying the cube using PROC SQL, you can generate a SAS dataset by creating a ‘slice’ from your Cube view. You may want to do this in order to a) preview the SAS dataset that will be generated by the MDX query to ensure that it captures the data how you intended or; b) create a dataset for any ad hoc analysis you are doing in Enterprise Guide.

- To create a slice of the view you are looking at, return to your Process Flow in the Project Designer View
- Right click on the Cube Node and select, Create Slice.

This ‘slice’ is essentially a Work SAS dataset.
Here is the resulting slice or SAS dataset:

![SAS Enterprise Guide](image)

Now that the cube data view is available as a dataset, it can be:

- renamed and referenced as a SAS dataset;
- used do any analysis using the built in Enterprise Guide tasks and wizards;
- joined to other datasets or slices;
- exported to Excel or other application;
- sent as an email attachment;

What if you wanted to capture the Cube View in the table format displayed in the Cube Viewer rather than as a SAS dataset? In this case you’d want to create a Report Snapshot.

- **To create a Report Snapshot** of the view you are looking at, return to your **Process Flow** in the **Project Designer View**
- **Right click** on the **Cube Node** and select, **Create Report Snapshot**.
The Report Snapshot is a static HTML copy of the Cube view and will open as a new tab in Enterprise Guide.

There are a number of ways you can customize and use this report snapshot. You can modify it by changing the page set up, adding headers and footers and changing the style template applied to the snapshot. You'll notice that a grid called SAS Reports has been added to your Enterprise Guide project (next to the Process Flow tab). By right clicking on the Report Snapshot icon that has been added the SAS Reports grid, you can Export the snapshot to other Windows applications, send the snapshot to an email recipient, rename or delete the snapshot and also update its properties.

Let's apply a different style template to this Report Snapshot.

- **Right click** on the Report Snapshot icon in your grid
- **Select** Properties

You'll notice that the style selected by default is **Seaside**.
Using the drop down list, scroll through the available styles and select serifprinter.

If you open up your Report Snapshot now, you'll notice that the style has changed to serifPrinter. (Ahem, it's not very attractive so let's change it back to Seaside).

The export functionality allows you to export the table and save it as Excel, Word or other Windows format files. You can also simply right click copy and paste the Report Snapshot table into Excel and preserve the formatting.
It’s important to remember that if your **Graph View** is still switched on in your Cube Viewer, the graph corresponding to your data view will also be captured in your Report Snapshot.

**OTHER CONSIDERATIONS – MICROSOFT CUBES**

In this paper, we’ve focused our attention on exploring and querying SAS OLAP cubes but it is important to mention that SAS OLAP Server is OLE DB compliant and can therefore access Microsoft SQL Server Cubes. You can explore a Microsoft cube using Enterprise Guide and you can also query it using PROC SQL via the SAS/ACCESS interface to OLE DB. Running PROC SETINIT will let you know if you have this product licensed in your environment. Here is the syntax required to connect to a Microsoft Cube using SQL Pass-through Facility:

```
Proc SQL;
Connect to OLEDB (provider= MSOLAP
 props=('DATA SOURCE’ = SQL Server Name
 ‘USER ID’ = User Id
 PASSWORD = Password
 ‘INITIAL CATALOGUE’ = CubeName ));
Create table MYDATA as select * from connection to OLEDB
( select <MDX query here>)
Quit;
Run;
```

There are other Properties (or PROPS) statements that may be required depending on your environment. For a complete description of all connection options as well as further discussion on querying Microsoft Cubes, please refer to the recommended reading links at the end of this paper.

**CONCLUSION**

This paper has shown how Enterprise Guide provides SAS Programmers with a method of exploring and querying OLAP Cubes. With the basics in place, you can now start exploring ways to build on this knowledge by looking at incorporating OLAP querying into your SAS Stored Processes, Enterprise Guide Projects or custom SAS applications. By applying your own SAS expertise, you can use OLAP cubes to build unique Reporting Solutions and tackle other data presentation requirements facing your organization.
RECOMMENDED READING
To gain a more complete understanding of Enterprise Guide 4.1 and using SAS OLAP Cubes, the author recommends the following web links.


ACKNOWLEDGMENTS

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

Rupinder Dhillon
Dhillon Consulting
Rupinder@dhillonconsulting.com
www.dhillonconsulting.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration. Other brand and product names are trademarks of their respective companies.
/* Macro Variables for use in Samples code */
%let SERVER = server.company.com; /* Metadata Server Name or IP Address */
%let PORT = 8561; /* Metadata Port Number */
%let PROTOCOL = Bridge; /* Metadata connection protocol */
%let USER = sasdemo; /* User ID with ReadMetadata and WriteMetadata permissions */
%let PASSWORD = sasdemo1; /* Password for the User Id */
%let REPOSITORY = Foundation; /* Repository in which to build data and cube */
%let LIBNAME = sample; /* Libref which references permanent data location */
%let LIBPATH = c:sasdata; /* Directory in which to store data */
%let SCHEMA = SASMain - OLAP Schema; /* OLAP Schema in which to store cube */
%let CUBEPATH = \Data; /* Physical location for OLAP Cube files */
%let CUBE = SampleCube ; /* Cube name */

/* Options for Metadata Server connections */
libname &libname "&libpath";

option metaserver="&SERVER"
   metaport=&PORT
   metaprotocol="&PROTOCOL"
   metarepository="&REPOSITORY"
   metauser="&USER"
   metapass="&PASSWORD";

data &libname..shoes;
    set sashelp.shoes;
run;

/* Register tables using PROC METALIB */
proc metalib;
   omr (library="&libname" metarepository="&repository");
   update_rule (delete); /* Overwrite dups regardless;
select (shoes);
report;
run;

/* Create the OLAP cube */
PROC OLAP
   Data=&libname..Shoes
   DrillThrough_Table=&libname..shoes
   cube=&cube
   Path="&cubepath"
   Description="Shoes Cube for HOW Examples"
;
METASVR repository="&repository" olap_schema="&schema"
   host="&server" port=&port;

DIMENSION Division hierarchies=(Division )
   CAPTION='Region'
   SORT_ORDER=ASCENDING ;

   HIERARCHY Division ALL_MEMBER='All Division'
   levels=( Region Subsidiary )
   CAPTION='Division'
   DEFAULT
   ;

LEVEL Subsidiary
CAPTION='Subsidiary'
SORT_ORDER=ASCENDING
;

LEVEL Region
CAPTION='Region'
SORT_ORDER=ASCENDING
;

DIMENSION Product_type hierarchies=(Product_type )
CAPTION='Product'
SORT_ORDER=ASCENDING ;

HIERARCHY Product_type ALL_MEMBER='All Product_type' levels=( Product )
CAPTION='Product'
DEFAULT
;

LEVEL Product
CAPTION='Product'
SORT_ORDER=ASCENDING
;

MEASURE ReturnsSUM
STAT=SUM
COLUMN=Returns
CAPTION='Sum of Returns'
FORMAT=DOLLAR12.
DEFAULT
;

MEASURE InventorySUM
STAT=SUM
COLUMN=Inventory
CAPTION='Sum of Inventory'
FORMAT=DOLLAR12.
;

MEASURE SalesAVG
STAT=AVG
COLUMN=Sales
CAPTION='Average Sales'
FORMAT=DOLLAR12.
;

MEASURE SalesSUM
STAT=SUM
COLUMN=Sales
CAPTION='Sum of Sales'
FORMAT=DOLLAR12.
;

MEASURE StoresSUM
STAT=SUM
COLUMN=Stores
CAPTION='Sum of Stores'
FORMAT=12.
;

AGGREGATION Region
    Subsidiary
Product
 / NAME='DEFAULT'
 ;

DEFINE
 MEMBER '[SampleCube].[Measures].[Sales per Store]' AS
   (([Measures].[SalesSUM] / [Measures].[StoresSUM]), FORMAT_STRING = "dollar30.");
RUN;

PROGRAM_2

/* Macro Variables for use in Samples code */
%let SERVER = server.company.com; /* Metadata Server Name or IP Address */
%let PORT = 8561; /* Metadata Port Number */
%let PROTOCOL = Bridge; /* Metadata connection protocol */
%let USER = sasdemo; /* User ID with ReadMetadata and WriteMetadata permissions */
%let PASSWORD = sasdemo1; /* Password for the User Id */
%let REPOSITORY = Foundation; /* Repository in which to build data and cube */
%let LIBNAME = sample; /* Libref which references permanent data location */
%let LIBPATH = c:\sasdata; /* Directory in which to store data */
%let SCHEMA = SASMain - OLAP Schema; /* OLAP Schema in which to store cube */
%let CUBEPATH = .\Data; /* Physical location for OLAP Cube files */
%let CUBE = SampleCube; /* Cube Name */
%let Olap_Port = 5451; /* OLAP Server Port */

Proc SQL;
Connec to OLAP (host=&server port=&olap_port protocol=&protocol user=&user
   pass=&password
   repository=&repository
   olap_schema=&schema);
Create table CanadaData as select * from connection to OLAP
   ( insert MDX query here )
Disconnect from OLAP;
Quit;

Ods pdf file = 'c:\ Sales in Canada.pdf';
Proc report data = CanadaData;
Title1 j=c "Shoe Sales in Canada";
Columns subsidiary StoresSUM SalesAVG;
Define subsidiary / order "subsidiary";
Define StoresSUM / display "Number of stores";
Define SalesAVG / sum "Average Sales";
brake after subsidiary / summarize;
run;
ODS PDF close;