

Paper 043-2011

How to Customize Your Data Analysis with SAS[®] OLAP Cube Calculations

Tatyana Petrova, SAS Institute Inc., Cary, NC

ABSTRACT

When a cube designer is working on an OLAP cube structure, several decisions should be made:

- Cube granularity
- Content and structure of dimensions
- What data facts from input tables are relevant to your analysis and what calculations do you need to apply to this data.

In this paper, we focus on the last decision. SAS OLAP Server provides various means to customize analysis of input data facts. Calculations can be created via:

- Cube Measures
- Cube-defined Calculated Measures and Members
- Global-scope/Session-scope Calculated Measures and Members
- Query-scope Calculated Measures and Members.

For each of these levels of definitions, various functions and statistics can be applied.

This paper is targeted toward OLAP cube designers and advanced OLAP cube users. It discusses how to design OLAP cube calculations, where to define them, what functions are available for your use, and how to embed more control into a flow of your MDX programs.

INTRODUCTION

This paper is organized into five sections:

1. Calculated Members and Measures, where we discuss types of Calculated Members and their basic definition.
2. Customizing your cubes with SAS BI products. SAS OLAP Cube Studio and SAS Enterprise Guide[®] are used to demonstrate how to create and work with Calculated Members using these products. Information about other products, such as SAS Data Integration Studio, SAS Add-in for Microsoft Office, SAS Information Map Studio, and SAS Web Report Studio is available in the appendix to this paper.
3. Calculated Members functionality, where you find information and examples on MDX functions and SAS functions available for your use.
4. Querying of Calculated Members with details about how to include Calculated Members in your reports and how to define the order of your calculations using Solve Order.
5. Typical usage section, which provides several examples of common solutions using OLAP cube Calculated Members.

Displays in this paper are taken from SAS 9.3 and SAS BI products of 4.3 release.

We use the PRDMDDDB cube for our code examples and displays. The PRDMDDDB cube has three dimensions and a set of measures (measures dimension) defined. Here is the structure of this cube:

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

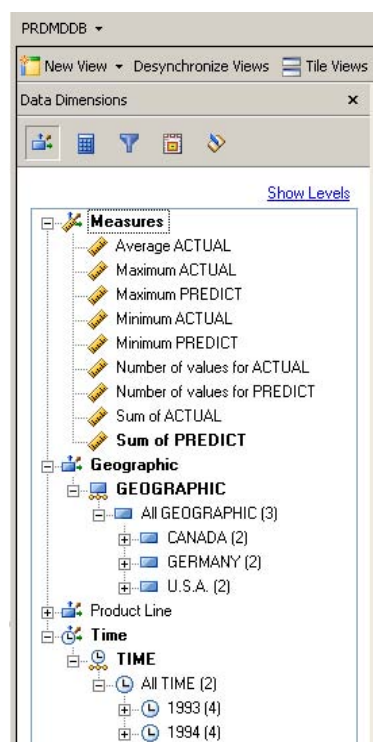


Figure 1: Structure of the PRDMDDB Cube as seen from SAS Enterprise Guide

CALCULATED MEMBERS, MEASURES, AND NAMED SETS

We begin with explaining what Cube Measures and Cube Calculated Members and Measures are. *Cube Measures* are based on underlying data columns with one of the supported statistics applied to the data facts from these columns. One data column can feed into multiple Cube Measures by applying different types of statistics.

In addition to OLAP built-in statistics, you can add your own calculations to the cube. These can be formulas that include already defined measures and cube members. The formulas can be built using MDX functions and operators as well as SAS functions. These custom calculations are called *Calculated Members*. They can be added as a member to any cube dimension, including measures. Most commonly, Calculated Members are added to a Measures dimension; thus, often in the BI world, you would see references to *Calculated Measures* to stress this fact. In SAS BI Products' interfaces, a Calculated Member term usually indicates that a calculation has been added on a dimension other than Measures. In this paper when the difference is not important, we refer to both Calculated Members and Measures as [CM].

CMs can be defined on various stages of a cube design and querying. The way you define a CM impacts the scope of it, that is, from where this CM can be accessed and by whom.

There are three types of scope for CMs:

- Global
- Session
- Query

And there are three ways to define CMs:

- Cube-defined (can define Global-scope CMs this way)
- Created with the MDX CREATE Statement (can define Global and Session-scope CMs this way)
- Created with MDX query (can define Query-scope CMs this way).

Next, we discuss each of these CM definitions and what the differences between the scopes are.

Cube-Defined CMs

Cube-defined Calculated Members and Measures are defined during a cube creation and are stored as a part of cube metadata. Only definitions of these members get created. The values for these CMs are not presummarized during

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

the cube build; they are generated during the querying time.

Cube-defined CMs are Global. This means they can be accessed the same way as the rest of the cube members and measures. Also, security can be set on these members the same way security is set for any other cube member. This provides accessibility of these members to all clients and sessions where the OLAP cube is viewed, and it enables user-based permissions to be set on these members.

Cube-defined CMs can be created using the same tools that are used for cube creation, for example, batch PROC OLAP code or a GUI interface available from SAS OLAP Cube Studio or SAS Data Integration Studio.

In PROC OLAP, the DEFINE statement is used for adding CMs. It can be issued during the cube build or after the cube has already been defined. In the latter case, it updates existing cube definition in the metadata. If issued later, the PROC OLAP METASVR information should be provided, and the DEFINE Statement should include the corresponding cube name.

Example of a cube-defined Calculated Measure with PROC OLAP syntax:

```
PROC OLAP;
  METASVR HOST=&host PORT=&port PROTOCOL=&protocol
  USERID=&userid PW=&pw REPOSITORY=&repos
  OLAP_SCHEMA=&olap_schema;

  DEFINE Member '[PRDMDDDB].[Measures].[Budget to Actual %]' AS
    '([Measures].[PREDICT_SUM]-[Measures].[ACTUAL_SUM]) /
    [Measures].[PREDICT_SUM], FORMAT_STRING="PERCENT6.1"';

RUN;
```

Example of a cube-defined Calculated Member built for other than measures dimension:

```
DEFINE Member '[PRDMDDDB].[GEOGRAPHIC].[All GEOGRAPHIC].[North America]' AS
  '[GEOGRAPHIC].[All GEOGRAPHIC]-[GEOGRAPHIC].[All GEOGRAPHIC].[GERMANY]';
```

Undefined a Cube-Defined CM

If you can define extra calculations for your cube, you should be able to undefine them. The PROC OLAP UNDEFINE statement enables you to do so:

```
UNDEFINE Member '[PRDMDDDB].[Measures].[Budget to Actual %]';
```

Global-Scope/Session-Scope Calculated Measures and Members Added with MDX CREATE Statements

In addition to cube-defined CMs, you can issue MDX statements to create Global or Session calculated members.

```
CREATE GLOBAL Member [PRDMDDDB].[Measures].[Budget to Actual %] AS
  '([Measures].[PREDICT_SUM]-[Measures].[ACTUAL_SUM]) /
  [Measures].[PREDICT_SUM], FORMAT_STRING="PERCENT6.1" '
or
```

```
CREATE SESSION Member [PRDMDDDB].[Measures].[Budget to Actual %] AS
  '([Measures].[PREDICT_SUM]-[Measures].[ACTUAL_SUM]) /
  [Measures].[PREDICT_SUM], FORMAT_STRING="PERCENT6.1" '
```

Note: if omitting the Global or Session keyword, the Session CM is created by default.

Information about Global CMs is stored as part of the cube metadata. To be able to create or drop Global CMs, you need to have WriteMetadata permission to the corresponding metadata repository and the cube itself.

Created with the MDX CREATE Statement, Global CM has the same scope and usage as a cube-defined CM. This CM becomes available to users with permission to access the cube as soon as the cube gets refreshed (issue the REFRESH CUBE administrative command on a server) or when the OLAP server gets restarted.

Adding a Global CM with an MDX statement might come in handy when different people or processes are responsible for Cube build versus customization. It also enables you to add or drop Global CMs without affecting the

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

cube build process.

Information about Session CMs is stored in session memory and does not get written to the metadata. Session CM is visible only within the scope of the running session or connection and is not saved for future access. As soon as you disconnect, you lose the definition of the Session CM. Other users cannot access this CM, and different users can create Session CMs with the same names.

Session CMs are great for one-time analysis or experiments with calculations before committing new CMs to the cube for global access.

Dropping Global or Session Member

The DROP command can be used to delete a Global CM's definition or a definition of a Session CM if needed before the session ends.

```
DROP Member [PRDMDDDB].[Measures].[Budget to Actual %]
```

Note: be extremely careful when using the DROP command. Be sure not to drop any global members that you did not intend to; this affects ALL the users of the cube.

Query Scope Calculated Measures and Members Created with MDX Query

Query-scope CMs are accessible only within the query with which they are defined. There is never a need to drop a Query CM.

Query CMs are most helpful when one-time analysis is needed for a cube or when different reports require different types of calculations, and the decision is made to store calculation definitions within the reports rather than in the cube. Another benefit is that the report designer or business analyst does not need to coordinate their calculation modifications with a cube designer and BI administrator. You also often see Query-scope Calculated Members generated automatically by the Reporting clients.

Example of a Query-scope CM definition:

```
WITH MEMBER [Measures].[Budget to Actual %] AS
    '([Measures].[PREDICT_SUM]-[Measures].[ACTUAL_SUM])/
    [Measures].[PREDICT_SUM], FORMAT_STRING = "PERCENT6.1"'
```

When deciding where to add your CM, consider the following:

- How many times will you need to redefine this CM if it's stored outside of a cube structure (that is, not a Global member)?
- Does the definition of a CM depend on a report or is it the same across reports or usage scenarios?
- Is there a chance that the user-based permission condition should be set on this CM or using this CM?
- Who is going to define this CM? Does this person have permission to create the CM within the needed scope?

Named Sets

Named Sets (sometimes called "Member sets") are aliases for set definitions. They are used to simplify MDX code and logic. The Named Set definition is executed once and is reused each time it's encountered. Named Sets, just like Calculated Members, can be defined with different scopes: global, session, and query.

Query scope:

```
WITH SET [MySet]
    AS 'Crossjoin({[GEOGRAPHIC].[All GEOGRAPHIC].Children},
    {[TIME].[All TIME].[1993].Children})'
SELECT
    [MySet] on columns,
    [Measures].[ACTUAL_SUM] on rows
FROM PRDMDDB
```

Global/Session Scope:

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

```
CREATE <GLOBAL> SET [PRDMDDDB].[MySet]
  AS 'CrossJoin({[GEOGRAPHIC].[All GEOGRAPHIC].Children},
    {[TIME].[All TIME].[1993].Children})'
```

The DROP SET command is available to drop a Global or a Session Named Set.

CUSTOMIZING YOUR CUBES WITH SAS BI PRODUCTS

In addition to defining Calculated Members in batch, you can use GUI wizards available in SAS BI products that support OLAP cubes.

SAS OLAP CUBE STUDIO

SAS OLAP Cube Studio can be used for creating and updating cube-defined (Global) CMs.

In SAS OLAP Cube Studio, select a cube that has already been defined. Right-click and select Maintain->Calculated Members...

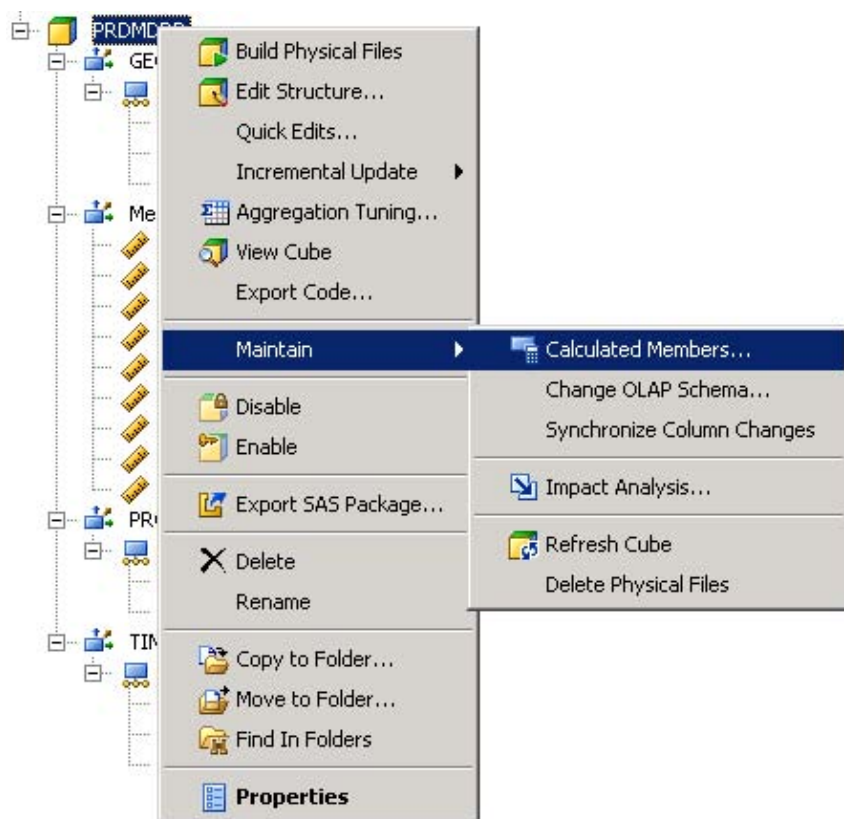


Figure 2: SAS OLAP Cube Studio. Invoking New Member wizard.

This brings you to the list of defined Calculated Members for the cube. You can Add, Edit, or Delete members from this list. Clicking Add, launches a New Member wizard, as shown in Figure 3.

We create similar Calculated Members like we did for the PROC OLAP examples, starting with [Measures].[Budget to Actual %]. The calculation type "Simple calculation" does fine as a calculation type for this new member.

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

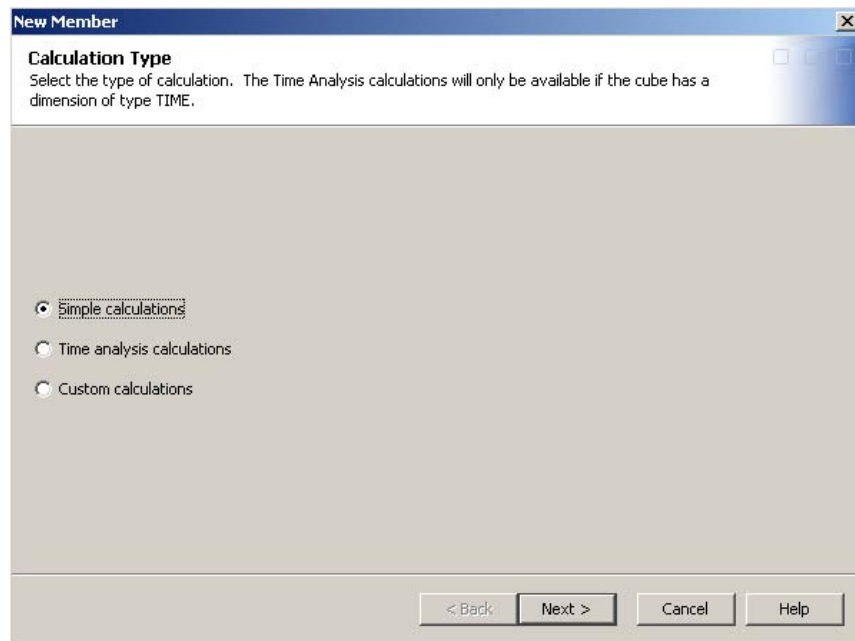


Figure 3: SAS OLAP Cube Studio. New Member wizard. Selecting calculation type.

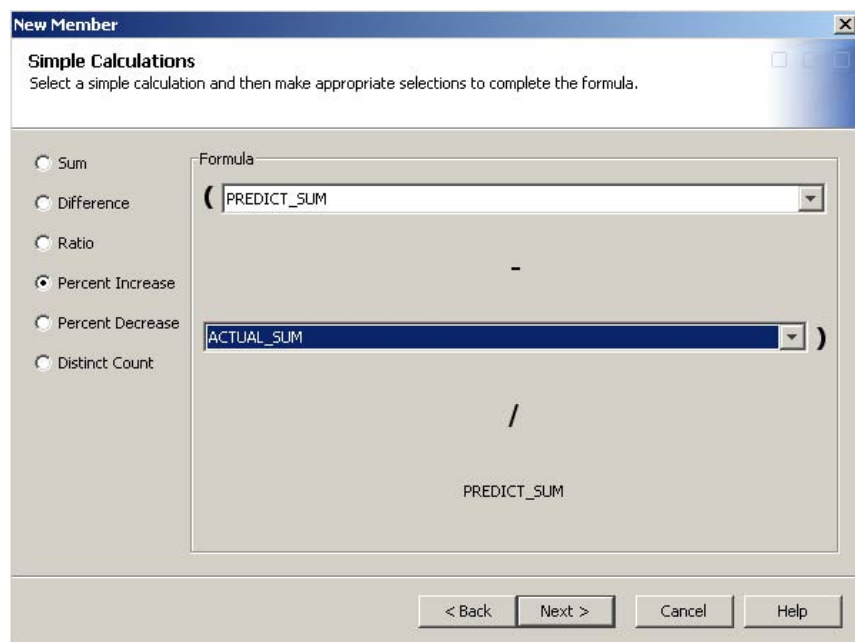
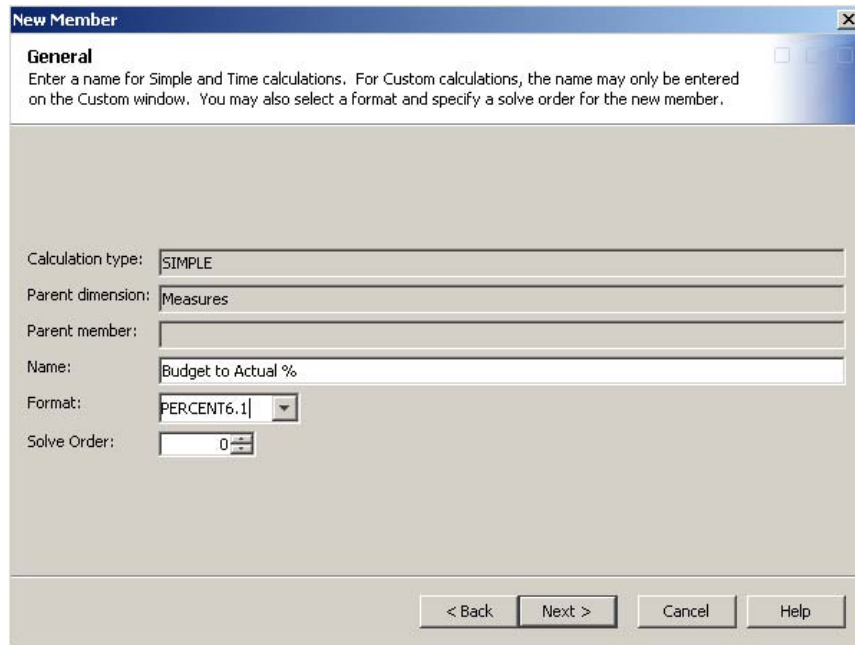


Figure 4: SAS OLAP Cube Studio. New Member wizard. Defining simple calculation for [Measures].[Budget to Actual %].

Next, general information about the new Calculated Member is collected, such as name, format and Solve Order (more about Solve Order in the section "Querying of Calculated Members.").

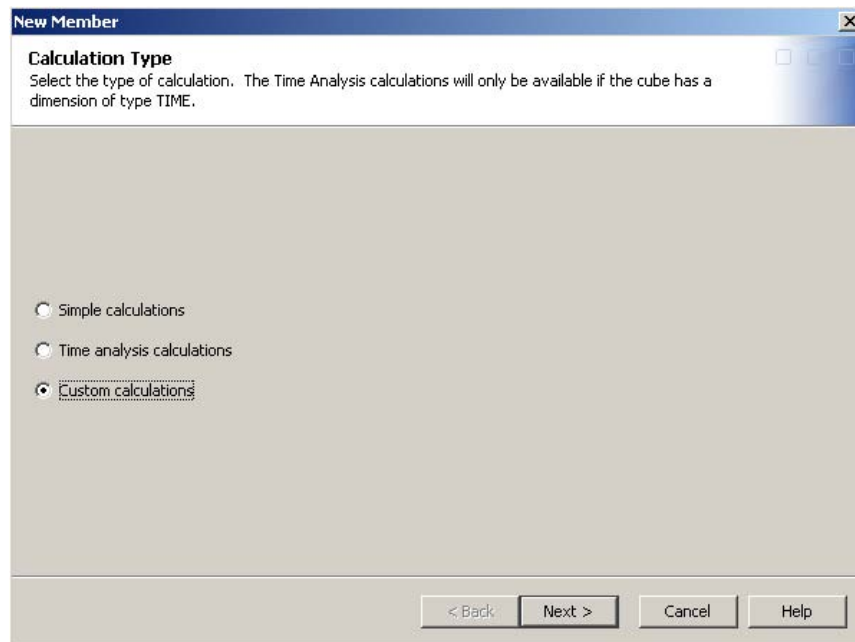
How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued



The screenshot shows the 'New Member' dialog box in SAS OLAP Cube Studio, specifically the 'General' tab. The dialog has a title bar with 'New Member' and a close button. Below the title bar is a header section with the title 'General' and a brief instruction: 'Enter a name for Simple and Time calculations. For Custom calculations, the name may only be entered on the Custom window. You may also select a format and specify a solve order for the new member.' The main area contains several input fields: 'Calculation type' is set to 'SIMPLE', 'Parent dimension' is 'Measures', 'Parent member' is empty, 'Name' is 'Budget to Actual %', 'Format' is 'PERCENT6.1' with a dropdown arrow, and 'Solve Order' is '0' with a spinner. At the bottom are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

Figure 5: SAS OLAP Cube Studio. New Member wizard. Providing general information about [Measures].[Budget to Actual %].

This completes [Measures].[Budget to Actual %] creation. Now we add a second Calculated Member [GEOGRAPHIC].[All GEOGRAPHIC].[North America]. We also could have used the “Simple calculations” wizard for this calculation, but we go with “Custom calculations” for demonstration purposes.



The screenshot shows the 'New Member' dialog box in SAS OLAP Cube Studio, specifically the 'Calculation Type' tab. The dialog has a title bar with 'New Member' and a close button. Below the title bar is a header section with the title 'Calculation Type' and a brief instruction: 'Select the type of calculation. The Time Analysis calculations will only be available if the cube has a dimension of type TIME.' The main area contains three radio button options: 'Simple calculations', 'Time analysis calculations', and 'Custom calculations'. The 'Custom calculations' option is selected. At the bottom are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

Figure 6: SAS OLAP Cube Studio. New Member wizard. Selecting type of calculation for [GEOGRAPHIC].[All GEOGRAPHIC].[North America].

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

The Custom calculation prompts you for general member information upfront. You can type in your formula on this screen or use Build Formula.

New Member

Custom Calculation
Enter a custom calculation. In addition to entering a formula for the custom calculation, you may select the parent dimension, parent member, name, format and solve order.

Parent dimension:

Parent member:

Name:

Format:

Solve Order:

Formula:

Figure 7: SAS OLAP Cube Studio. New Member wizard. Custom calculation.

Selecting Build Formula brings you into the interface for MDX expression creation.

Build Formula

Expression Text:

Data Elements:

- PRDMDDB
 - Geographic
 - GEOGRAPHIC
 - All GEOGRAPHIC (selected)
 - CANADA
 - GERMANY
 - U.S.A.
 - Product Line
 - Time
 - Measures
 - Member Properties

Figure 8: SAS OLAP Cube Studio. New Member wizard. Build Formula interface. Defining MDX expression for [GEOGRAPHIC].[All GEOGRAPHIC].[North America].

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

The Functions tab (See Figure 8.) provides a list of MDX functions with functions syntax help. The Data Sources tab allows access to the cube members' structure from which to select members. The Build Formula interface provides a mechanism for validating your MDX expression. You need a running OLAP server to be able to use it. Clicking the OK button also triggers the expression validation. You cannot save an invalidated formula.

At this point both Calculated Members are successfully created. If you go to any reporting tool that works with OLAP cubes, you are able to see and query these two new Calculated Members.

For example, here is a view of the cube structure in SAS Enterprise Guide:

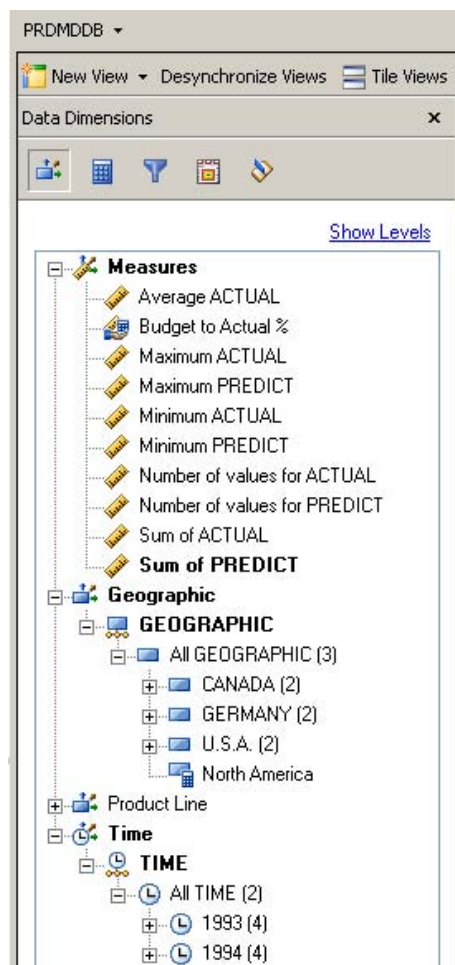


Figure 9: Structure of the PRDMDDDB Cube after two Calculated Members have been added, as seen from SAS Enterprise Guide.

SAS ENTERPRISE GUIDE

SAS Enterprise Guide allows creation of Global, Session, and Query-scope Calculated Members; however, there is limited control over Global Calculated Members – see details below.

When defining a Global or Session CM, you can use a set of wizards that walk you through a CM definition process. Behind the scenes, SAS Enterprise Guide issues an MDX CREATE Statement.

To add Global or Session Calculated Members in SAS Enterprise Guide, go to an OLAP Viewer->Customized Items and Sets tab (See Figure 10.).

The “New” button provides you access to the creation of Calculated Measures, Calculated Members, and Member Sets (Named sets).

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

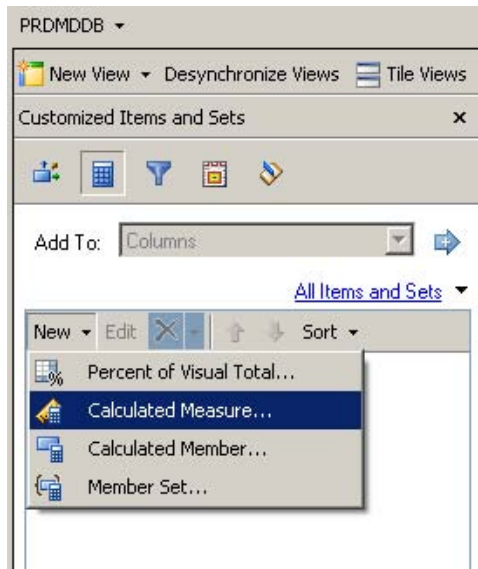


Figure 10: SAS Enterprise Guide. Customized Items and Sets pane.

This brings you to a wizard similar to the one in SAS OLAP Cube Studio, with options available for creating Basic Analysis and Special Analysis. Several types of calculations are pre-built in SAS Enterprise Guide, like Time Series calculations (see Figure 11 as an example of what functionality is available for Time Series Analysis) and Custom Analysis (type in MDX expression or use the Expression Builder, which allows drag and drop capabilities and context help for MDX functions syntax).

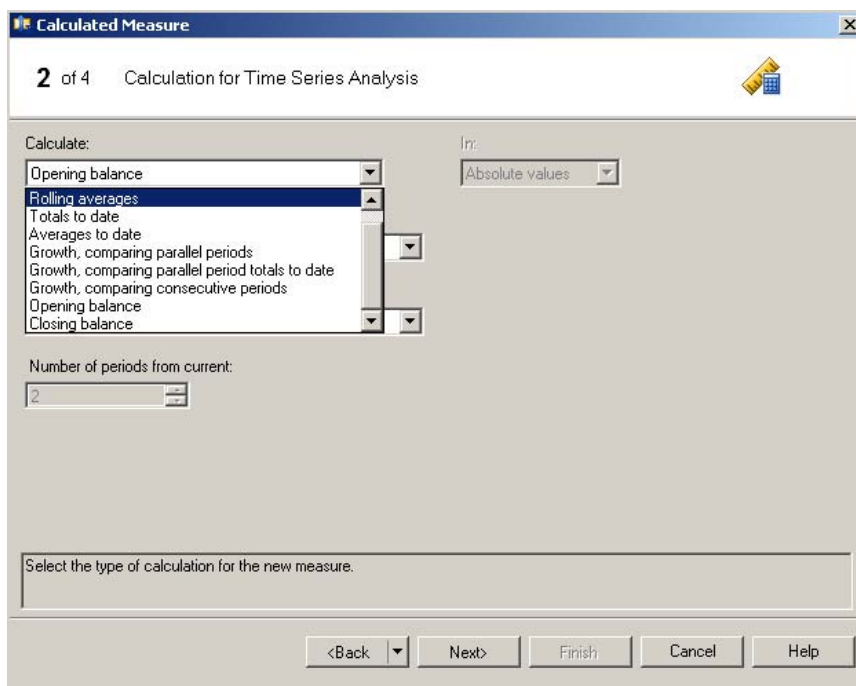


Figure 11: SAS Enterprise Guide. Calculated Measure wizard. Pre-built calculations for Time Series Analysis.

On the next step, you are provided with options for choosing a scope of your CM.

You can declare your CM to be Session (option "The new measure is available: Locally, to me") or Global ("The new measure is available: Publicly, to all users at all times (you are no longer able to edit the item)").

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

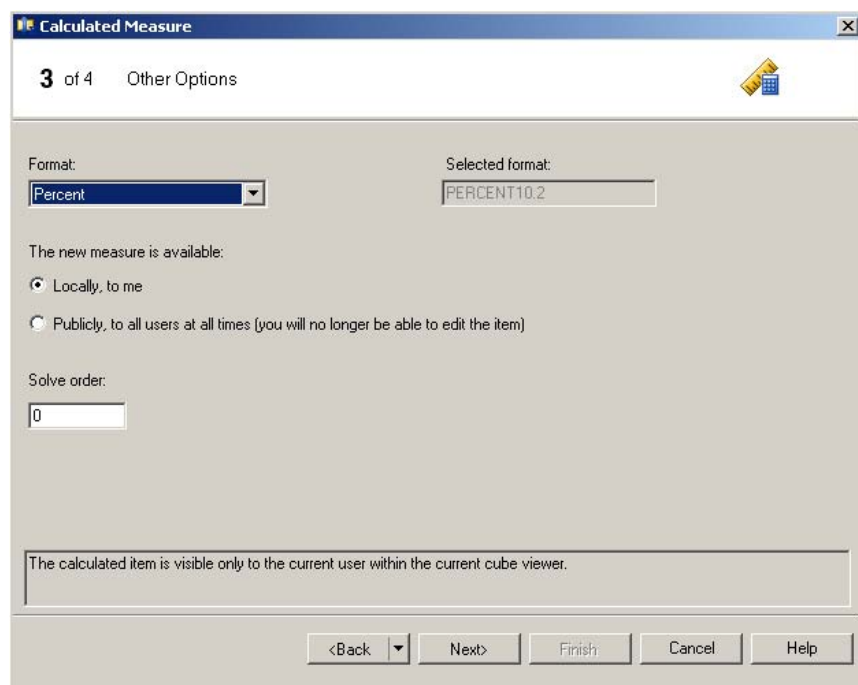


Figure 12: SAS Enterprise Guide. Calculated Measure wizard. Window to set general options for a Calculated Measure, such as Format and Solve Order.

As with any Global changes, it is recommended that you first experiment with a new calculation locally and then commit it to Public, by changing the scope property of the Calculated Member or Measure. Notice that you cannot edit or delete Public CM via this interface.

Note: If you do need to update or drop a newly added Public CM, you have an option to either contact your BI administrator or go to a different tool, such as SAS OLAP Cube Studio. You can also use a SAS Enterprise Guide built-in MDX Editor to issue an explicit DROP command for this member.

You can use SAS Enterprise Guide to work with Query-scoped calculations as well. The built-in MDX Editor provides an interface to create or update MDX queries for the current report, including Calculated Member specifications. To invoke the MDX Editor, go to "Edit View" menu on the toolbar -> Edit with MDX Editor. (See Figure 13.)

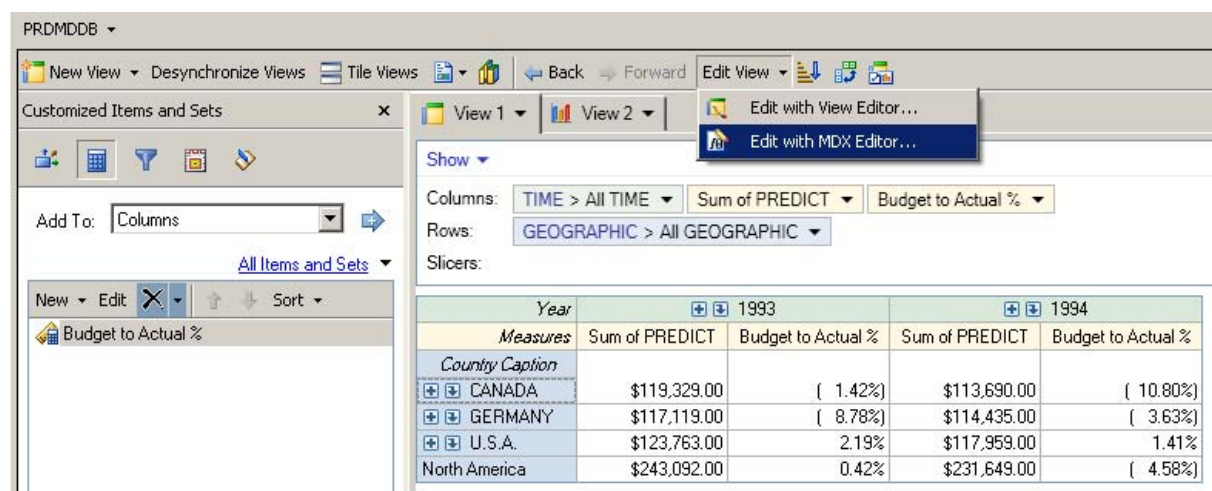


Figure 13: SAS Enterprise Guide. Invoking MDX Editor.

In this figure, the report displays values for a Session-scope CM [Budget to Actual %] (also listed as a CM on a Customized Items and Sets tab) and a Query-scope CM [North America].

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

Information about how to work with Calculated Members in SAS Data Integration Studio, SAS Add-in for Microsoft Office, SAS Information Map Studio, and SAS Web Report Studio is available in the Appendix to this paper.

CALCULATED MEMBERS FUNCTIONALITY

MDX Functions

SAS supports Standard MDX, including many MDX functions (refer to “SAS 9.2 OLAP Server: MDX Guide” for information about MDX functions). SAS also provides a set of MDX extensions that increase flexibility of SAS MDX, such as support for SAS functions and several others.

One of the examples of MDX extensions added by SAS is the ISMISSING function. ISMISSING is similar to the Standard MDX ISEMPTY function, but supports filtering of missing values instead of empty. It returns 1 if the value passed in is missing and 0 otherwise. A typical example where the ISMISSING function is used is assigning a special value during calculations over cells with missing values.

Example on handling empty or missing values:

```
WITH
MEMBER [Measures].[Actual Price] AS
    'IIf (
        isEmpty([Measures].[ACTUAL_N]), NULL,
        IIf(isMissing([Measures].[ACTUAL_N]), -1,
            [Measures].[ACTUAL_SUM] / [Measures].[ACTUAL_N] * 100
        )
    ) '
```

SAS Functions

When building MDX expressions, in addition to MDX functions, you can use all SAS functions that are relevant in MDX context including a set of SAS date functions, numeric calculations and operations with strings (refer to *SAS 9.2 Language Reference: Dictionary, Fourth Edition* for information about SAS functions).

Here is an example of using a logical SAS function IFN, which conveniently allows you to avoid extra coding for missing values.

Syntax:

```
IFN(logical-expression, value-returned-when-true, value-returned-when-false <,value-returned-when-missing>)
```

Example on how handling empty or missing values formula can be rewritten with the IFN function:

```
WITH
MEMBER [Measures].[Actual Price] AS
    'IFN (
        isEmpty([Measures].[ACTUAL_N]), NULL,
        [Measures].[ACTUAL_SUM] / [Measures].[ACTUAL_N] * 100, -1
    ) ',
    FORMAT_STRING="DOLLAR12.2"
```

Note: Cell formats are not propagated through calculations if SAS functions are used within the formula. Use explicit formats assignment for the Calculated Member for such cases.

There are a number of functions that have the same names in Base SAS as in MDX. For example, the SUM function is one of them. Functionality of the SUM function is different in MDX than in Base SAS. The Base SAS version `<SUM(argument,argument,..)>` returns a sum of listed nonmissing arguments. The MDX version `<SUM(<Set>[,<Numeric Expression>]>` returns a sum of values across a set of tuples using the numeric value expression (usually some measure) as a base for aggregation. If no numeric value is specified, the SUM function uses the current context (that is, current members that are not specified in the set dimensions).

You might want your function to behave one way or another. If Base SAS function's version is preferable, specify the

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

SAS! prefix to distinguish it from the MDX function: SAS!SUM(<parameters>).

SetToList

Another useful MDX extension is the SetToList function. It allows SAS functions to operate over MDX sets while not explicitly listing all the set members.

Imagine that you want to see the 85th percentile of the children of a given Geographic member and a given Time member for a certain measure. To calculate the percentile, you can use the SAS function

PCTL:PCTL<n>(percentage, value1<, value2, ...>)

We obviously do not want to list all the members of the set from Geographic and Time dimensions as value1, value2, and so on. Instead, we can use the MDX function SetToList. SetToList takes an MDX set as an input parameter and returns a list of comma-separated values based on a Numeric or Character expression. Then a SAS function can use this list as input parameters.

```
SetToList(<Set>[,<Numeric Expression | Character Expression>])
```

Example:

```
WITH
MEMBER [Measures].[Percentile85] AS
    'PCTL5(
        85,SetToList(
            {CrossJoin({[GEOGRAPHIC].CurrentMember.Children},
                {[TIME].CurrentMember.Children})),
            [Measures].[ACTUAL_SUM])',
        FORMAT_STRING="DOLLAR12.2"
    '

SET [MySet] AS
    'CrossJoin({[GEOGRAPHIC].[All GEOGRAPHIC].Children},
        {[TIME].[All TIME].[1993].Children})'

SELECT
{[Measures].[ACTUAL_SUM],
    [Measures].[Percentile85]} on columns,
[MySet] on rows
FROM PRDMDDB
```

Results:

<i>Measures</i>		Sum of ACTUAL	Percentile85
<i>Country Caption</i>	<i>Quarter</i>		
CANADA	1	\$29,004.00	\$5,438.00
	2	\$30,815.00	\$6,089.00
	3	\$31,046.00	\$6,525.00
	4	\$30,155.00	\$6,011.00
GERMANY	1	\$30,308.00	\$5,936.00
	2	\$34,880.00	\$7,167.00
	3	\$32,633.00	\$7,238.00
	4	\$29,583.00	\$6,487.00
U.S.A.	1	\$29,958.00	\$5,852.00
	2	\$30,108.00	\$5,961.00
	3	\$29,755.00	\$6,744.00
	4	\$31,232.00	\$6,309.00

QUERYING OF CALCULATED MEMBERS

HOW TO RETRIEVE

To retrieve Calculated Members of any scope in your query, you either need to reference them explicitly by name or

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

use one of these two functions:

- `.AllMembers` - available in `dimension.AllMembers`, `hierarchy.AllMembers`, `level.AllMembers` form.

Use this function when you would normally use the `.Members` function to retrieve all members, including calculated members of a specified dimension/hierarchy/level. A regular `.Member` function does not return OLAP cube calculated members.

For example,

```
[Measures].AllMembers
```

- `AddCalculatedMembers (set)` – this function returns *set* plus all calculated members that are siblings of the members specified within the *set*.

For example,

```
AddCalculatedMembers ([GEOGRAPHIC].[All GEOGRAPHIC].Children)
```

SAS Enterprise Guide and SAS Add-in for Microsoft Office enable you to set an option to “Show calculated members in results.” This option can be found in View Properties of the OLAP Viewer (Right-click anywhere within your Results view) -> Optimize Results tab. This option is set OFF by default. If you set it ON, you get `AddCalculatedMembers()` added to your queries.

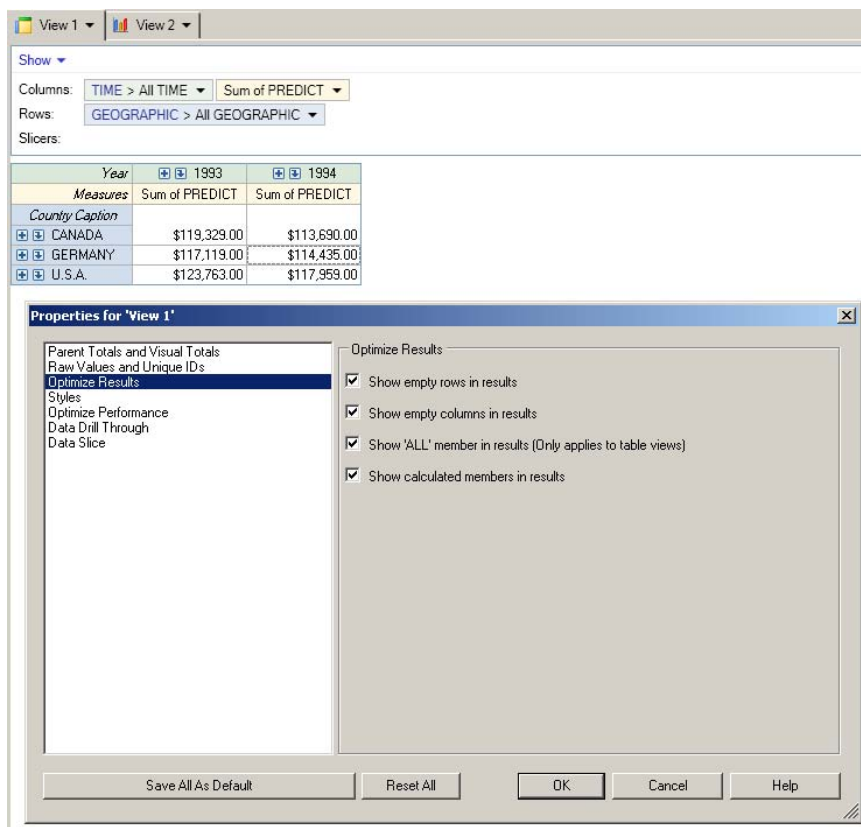


Figure 14: SAS Enterprise Guide. Setting an option to show calculated members in results of the query.

Or you can set this option permanently.

In SAS Enterprise Guide, go to Tools->Options->OLAP Data->Other View options-> Optimize Results tab.

In SAS Add-in for Microsoft Office, in the SAS pane, go to Tools->Options->Advanced tab->OLAP Viewer Options-> Optimize Results tab.

This brings you to the same options interface as shown on Figure 14.

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

Depending on the coding scenario, sometimes it is helpful to be able to remove calculated members from the set. The StripCalculatedMembers(*set*) function enables you to do exactly that.

ORDER OF CALCULATIONS

There are cases when an order of calculations is important for the end result.

You have an option to control in which order your calculations should be executed: SOLVE_ORDER=<value>.

You can either explicitly add this option to the end of your CM definition:

```
WITH
MEMBER [Measures].[Budget to Actual %] AS
    '([Measures].[PREDICT_SUM]-[Measures].[ACTUAL_SUM])/
      [Measures].[PREDICT_SUM]
    , FORMAT_STRING = "PERCENT6.1"', SOLVE_ORDER=2
```

Or provide a value for Solve Order within the New Member wizard when creating your CMs using a GUI interface. (See Figures 5, 7, 12 earlier in the paper for an example.)

Example of a query where order of calculations is important:

```
WITH MEMBER [Measures].[Actual Price] AS
    '[Measures].[ACTUAL_SUM]/[Measures].[ACTUAL_N]',
    SOLVE_ORDER=1

MEMBER [GEOGRAPHIC].[All GEOGRAPHIC].[North America] AS
    '[GEOGRAPHIC].[All GEOGRAPHIC]-[GEOGRAPHIC].[All GEOGRAPHIC].[GERMANY]',
    SOLVE_ORDER=2

SELECT
    {[GEOGRAPHIC].[All GEOGRAPHIC],
     [GEOGRAPHIC].[All GEOGRAPHIC].[North America]} on columns,
    {[Measures].[ACTUAL_SUM],
     [Measures].[ACTUAL_N],
     [Measures].[Actual Price]} on rows
FROM PRMDDB
```

Results:

<i>All GEOGRAPHIC</i>	All GEOGRAPHIC	
<i>Country Caption</i>		North America
<i>Measures</i>		
Sum of ACTUAL	\$730,337.00	\$484,339.00
Number of values for ACTUAL	1440	960
Actual Price	\$507.18	\$-5.32

The results of crossing [Actual price] with [North America] are **not** what you want to see. This cell returns a negative price because of the order of calculations - you get the difference of Actual Price instead of the quotient of measures for North America.

Instead, swap the order and get more meaningful results, for example:

```
WITH MEMBER [Measures].[Actual Price] AS
    '[Measures].[ACTUAL_SUM]/[Measures].[ACTUAL_N]',
    SOLVE_ORDER=2

MEMBER [GEOGRAPHIC].[All GEOGRAPHIC].[North America] AS
    '[GEOGRAPHIC].[All GEOGRAPHIC]-[GEOGRAPHIC].[All GEOGRAPHIC].[GERMANY]',
    SOLVE_ORDER=1
```


How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

Results:

<i>All GEOGRAPHIC</i>	All GEOGRAPHIC	
<i>Country Caption</i>		North America
<i>Measures</i>		
Sum of ACTUAL	\$730,337.00	\$484,339.00
Number of values for ACTUAL	1440	960
Actual Price	\$507.18	\$504.52

Think ahead when planning your calculated members. Leave gaps when specifying SOLVE_ORDER values in case you need to add more calculations with 'in-between' values later on.

PERFORMANCE IMPACT

Be careful when adding calculations to your cube. Calculated Members are not pre-aggregated. All the CM's values are calculated on-the-fly during the queries execution.

Depending on the types of calculations and the size of your data, Calculated Members processing might significantly impact your query performance. Be especially careful with sets manipulations (using functions like Filter() and Order()) and coding calculations that trigger an excessive number of subqueries to be executed.

TYPICAL USAGE

There is a broad range of Calculated Members and Measures applications. Some of the usage scenarios are very common and are used to solve the analysis tasks that are typical to every organization. Some are complicated scenarios that involve sophisticated MDX coding that solves unique analysis needs or serve as workarounds for any shortages in data structure, data content, or technologies used. Strong MDX expertise is needed for customization of this depth.

In addition to examples already shown in previous scenarios, here we provide a few more examples of the common OLAP analysis tasks that are solvable with OLAP cube Calculated Members.

1. Obtaining Comparative Results and Ratios Over the Period of Time

1.1. With Parallel Period function:

Calculate a measure that displays the difference of the Actual Price value of the current month and the value from last Quarter month (3 months ago):

```
WITH MEMBER [Measures].[Actual Price] AS
    '[Measures].[ACTUAL_SUM]/[Measures].[ACTUAL_N]'

MEMBER [Measures].[Actual Price Difference] AS
    'IIF(
        ([Measures].[Actual Price],
         ParallelPeriod([TIME].[MONTH],3,[TIME].CurrentMember)
        ) = NULL,
        NULL,
        ([Measures].[Actual Price], [TIME].CurrentMember)-
        ([Measures].[Actual Price],
         ParallelPeriod([TIME].[MONTH],3,[TIME].CurrentMember))
    )'

SELECT
    {[TIME].[All TIME].[1993].[1].[Jan]:[TIME].[All TIME].[1993].[2].[Jun]}
    on columns,
    {[Measures].[ACTUAL_SUM],
     [Measures].[ACTUAL_N],
     [Measures].[Actual Price],
```

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

```
[Measures].[Actual Price Difference]] on rows
FROM PRDMDDB
```

Note that the calculation accounts for cases when previous members do not exist and sets the Difference value to Null.

Results:

<i>Month</i>	Jan	Feb	Mar	Apr	May	Jun
<i>Measures</i>						
Sum of ACTUAL	\$29,813.00	\$29,584.00	\$29,873.00	\$30,581.00	\$31,617.00	\$33,605.00
Number of values for ACTUAL	60	60	60	60	60	60
Actual Price	\$496.88	\$493.07	\$497.88	\$509.68	\$526.95	\$560.08
Actual Price Difference	.	.	.	\$12.80	\$33.88	\$62.20

1.2. With PeriodsToDate function:

Aggregate values over the set of months for the current year, from the beginning of the year through the current month:

```
WITH
MEMBER [Measures].[Actual Sum to Date] AS
    'Aggregate(
        PeriodsToDate([Time].[YEAR], [Time].CurrentMember),
        [Measures].[ACTUAL_SUM]
    )'
SELECT
    {[TIME].[All TIME].[1993].[1].[Jan]:[TIME].[All TIME].[1993].[1].[Mar]}
    on columns,
    {[Measures].[ACTUAL_SUM],
     [Measures].[Actual Sum to Date]} on rows
FROM PRDMDDB
```

Results:

<i>Month</i>	Jan	Feb	Mar
<i>Measures</i>			
Sum of ACTUAL	\$29,813.00	\$29,584.00	\$29,873.00
Actual Sum to Date	\$29,813.00	\$59,397.00	\$89,270.00

2. Moving Averages

The task is to calculate an average of the last 6 months for each Time member from a given range.

There are multiple solutions for calculating moving (sometimes called “rolling”) averages. The choice of the function depends on how you want your Time periods to be defined.

In the simplest form, you can use the LastPeriods function to calculate the average of the last 6 months:

```
WITH MEMBER [Measures].[Avg for Periods to Date] AS
    'Avg(LastPeriods(6, [Time].CurrentMember), [Measures].[ACTUAL_SUM])'
SELECT
    {[TIME].[All TIME].[1993].[1].[Jan]:[TIME].[All TIME].[1993].[2].[Jun]} on columns,
    {[Measures].[ACTUAL_SUM],
     [Measures].[Avg for Periods to Date]} on rows
FROM PRDMDDB
```

Results:

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

Month	Jan	Feb	Mar	Apr	May	Jun
Measures						
Sum of ACTUAL	\$29,813.00	\$29,584.00	\$29,873.00	\$30,581.00	\$31,617.00	\$33,605.00
Avg for Periods to Date	\$29,813.00	\$29,698.50	\$29,756.67	\$29,962.75	\$30,293.60	\$30,845.50

But if the task was, for example, to calculate an average of the months between the current month and a similar month 2 quarters ago, instead of calculating the total of months to go back to be used in LastPeriods(), the ParallelPeriod function will come in more handy:

```
MEMBER [Measures].[Avg for Periods to Date] AS
    'Avg(
        {ParallelPeriod([Time].[Quarter], 2, [Time].CurrentMember) :
          [Time].CurrentMember}, [Measures].[ACTUAL_SUM]
        ) '
```

Another option is to use Lag function:

```
MEMBER [Measures].[Avg for Periods to Date] AS
    'Avg(
        Time.CurrentMember.Lag(6) : Time.CurrentMember,
        Measures.[Measures].[ACTUAL_SUM]
    ) '
```

Same as in example 1.2, when defining calculations that require backing up several members, be careful on how you want cases with nonexistent members to be handled. You might need to add additional logic that accounts for such cases.

You can also add some intelligence to your CM definition to be flexible depending on what the level of [Time].CurrentMember is: if Month, go back X number of periods; if Quarter, go back Y number of periods, and so on.

3. Weighted Statistics

A weight variable enables you to specify the relative importance of each observation. With no weight variable, all input records have a weight of 1. If you specify a weight of 2, that input record is counted as twice its value.

To implement weighted statistics with an OLAP cube, your data should have a column that stores the weight of each observation (column weightvar). Then in your cube, add the summed weight as a measure:

```
MEASURE Weight_sum COLUMN=weightvar STAT=sum;
MEASURE Actual_sum COLUMN=actual STAT=sum;
```

In MDX, create a Calculated Member:

```
WITH MEMBER [Measures].[Actual_Wsum] AS
    '[Measures].[Actual_sum] * [Measures].[Weight_sum] '
```

[Measures].[Actual_Wsum] represents your weighted values.

CONCLUSION

Calculated Members are a powerful tool available for OLAP cubes customization. It allows adding much more complex calculations for your OLAP analysis than the basic cube statistics do.

With power comes responsibility. Be careful working in Global calculations scope: all the changes that you make to public Calculated Members are visible to all users of the cube (unless you define extra security logic for these members). If hesitating, try them on a Query or Session level of the scope first.

Strong MDX expertise might be required if engaging in complicated customizations of your OLAP cube. SAS BI products provide a useful set of wizards to help guide basic and some of the most typical formulas to create. For more extensive coding, MDX Expression Builder provides MDX functions syntax help and access to cube members structure, as well as a validation mechanism for the code defined.

REFERENCES

SAS Institute Inc. 2009. *SAS® 9.2 OLAP Server: MDX Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2009. *SAS® 9.2 OLAP Server: User's Guide*. Cary, NC: SAS Institute Inc.

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

SAS Institute Inc. 2011. *SAS® 9.2 Language Reference: Dictionary, Fourth Edition*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Acknowledgments go to Matthias Ender, SAS Institute Inc., for a big help compiling information about user scenarios and examples, and to Mike Huels, SAS Institute Inc., for his careful review of this paper.

RECOMMENDED READING

Fast Track to MDX, Second Edition by M. Whitehorn, R. Zare and M. Pasumansky. 2005. Springer

MDX Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase, Second Edition, by G.Spofford, S.Harinath, C.Webb, D.Hai Huang, F.Civardi. 2006. John Wiley & Sons.

CONTACT INFORMATION

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

Tatyana Petrova
SAS Campus Drive
SAS Institute Inc.
E-mail: Tatyana.Petrova@sas.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.

APPENDIX: CUSTOMIZING YOUR CUBES WITH SAS BI PRODUCTS - CONTINUED

SAS ADD-IN FOR MICROSOFT OFFICE

In SAS Add-in for Microsoft Office, scope specification and creation of Calculated Members is consistent with SAS Enterprise Guide. You can define CMs of Public (Global) or Local (Session) scope: use Customized Items and Sets tab for adding Global or Session CMs. You can also use an MDX Editor for adding Query-scope calculations.

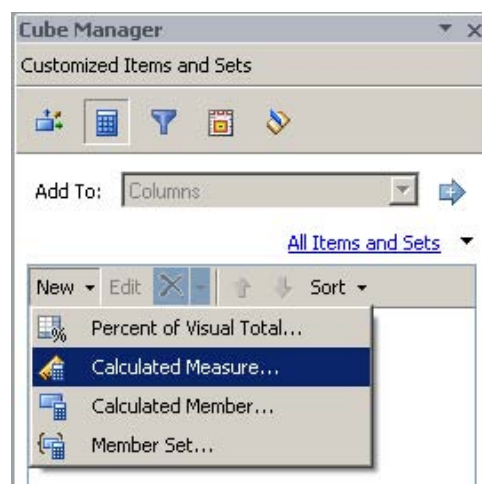


Figure 15: SAS Add-in for Microsoft Office. Invoking the Calculated Measure wizard.

SAS DATA INTEGRATION STUDIO

The interface for Cube maintenance via SAS Data Integration Studio, including creation of Calculated Members, is consistent with SAS OLAP Cube Studio. Refer to the section on SAS OLAP Cube Studio earlier in the paper to guide

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

you through.

SAS INFORMATION MAP STUDIO

When creating an Information Map, you can add Data Items to it that are either based on the data coming straight from a data source, or calculated based on data coming from a data source. If an OLAP cube is used as a data source, the later choice becomes an analog to an OLAP cube Calculated Member. Added data items will be available for any report that uses this Information Map, but they have no effect on the cube itself.

To invoke the New Data Item specification wizard, Click Insert on the SAS Information Map Studio toolbar.

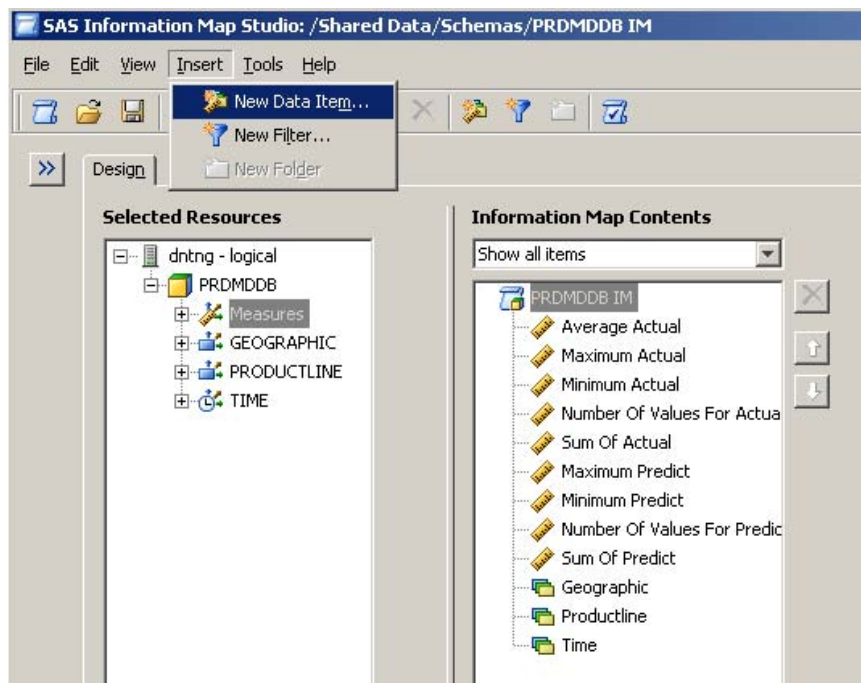


Figure 16: SAS Information Map Studio. Adding a New Data Item to an Information Map.

Here you can provide general information about your Data Item. Notice the option to include the Data Item in the default query.

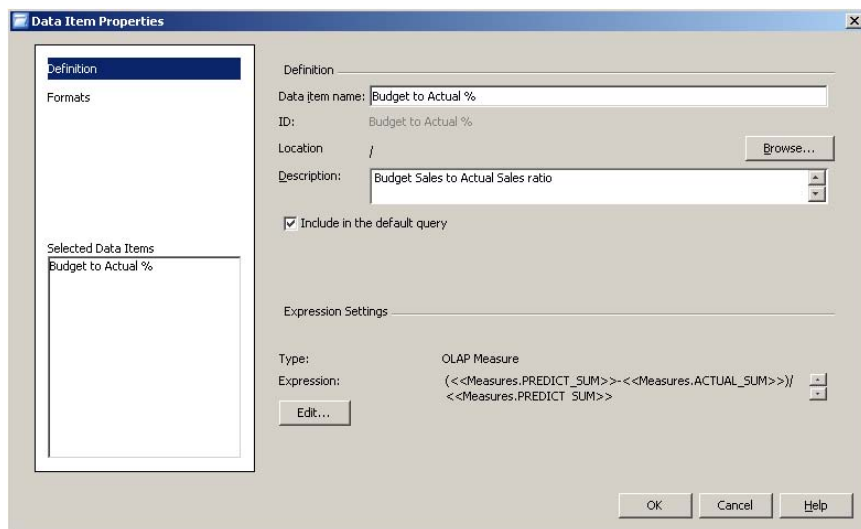


Figure 17: SAS Information Map Studio. Definition of New Data item “Budget to Actual %”.

The Edit button on this screen brings you to the familiar interface of the Expression Editor.

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

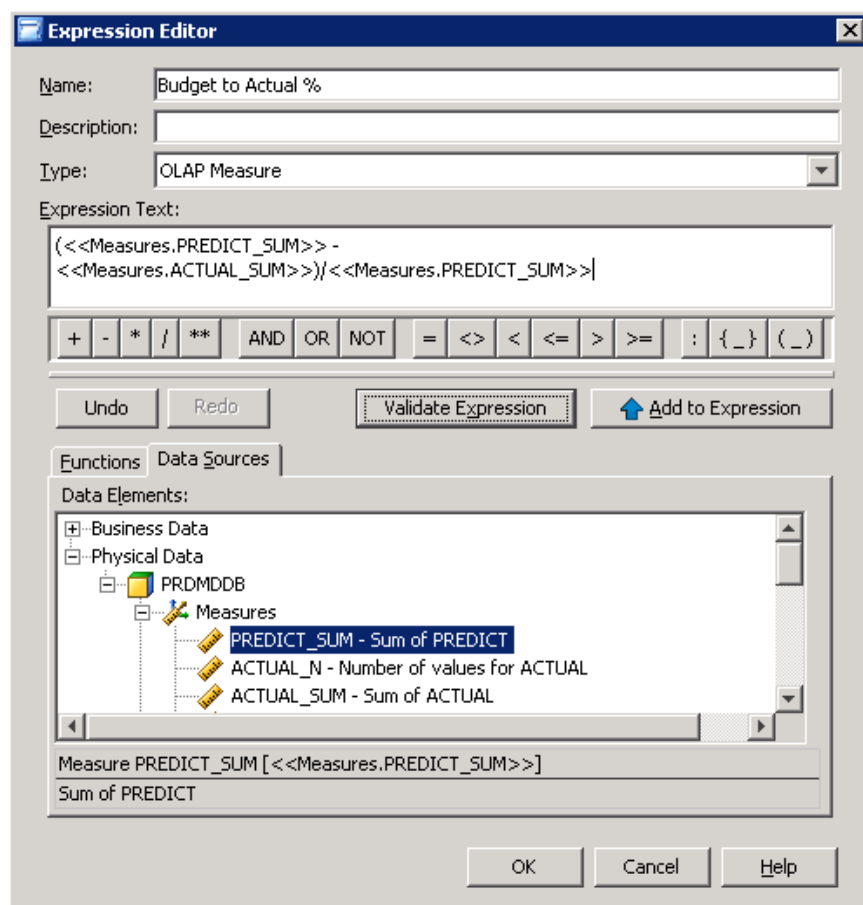


Figure 18: SAS Information Map Studio. Expression Editor for New Data Item.

Set Type to “OLAP Measure” to create an Information Map analog of a Calculated Measure or to “OLAP Category” for a Calculated Member.

SAS WEB REPORT STUDIO

SAS Web Report Studio enables you to create report-level calculations. These added Calculations have no effect on a cube or an information Map that serves data into the report, but it is available to all users who have access to this Web report.

SAS Web Report Studio provides a built-in Expression Builder for the simplest calculations (available from Options-> Select Data).

How to Customize Your Data Analysis with SAS® OLAP Cube Calculations, continued

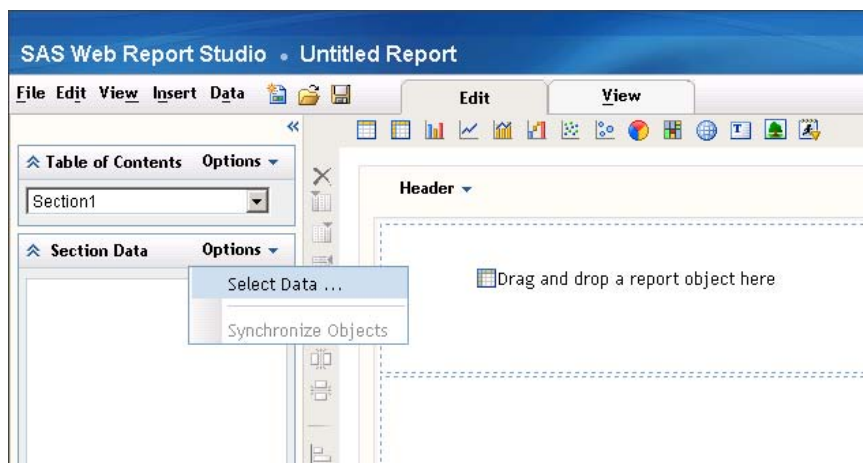


Figure 19: SAS Web Report Studio. Invoking Select Data

Use Custom tab.

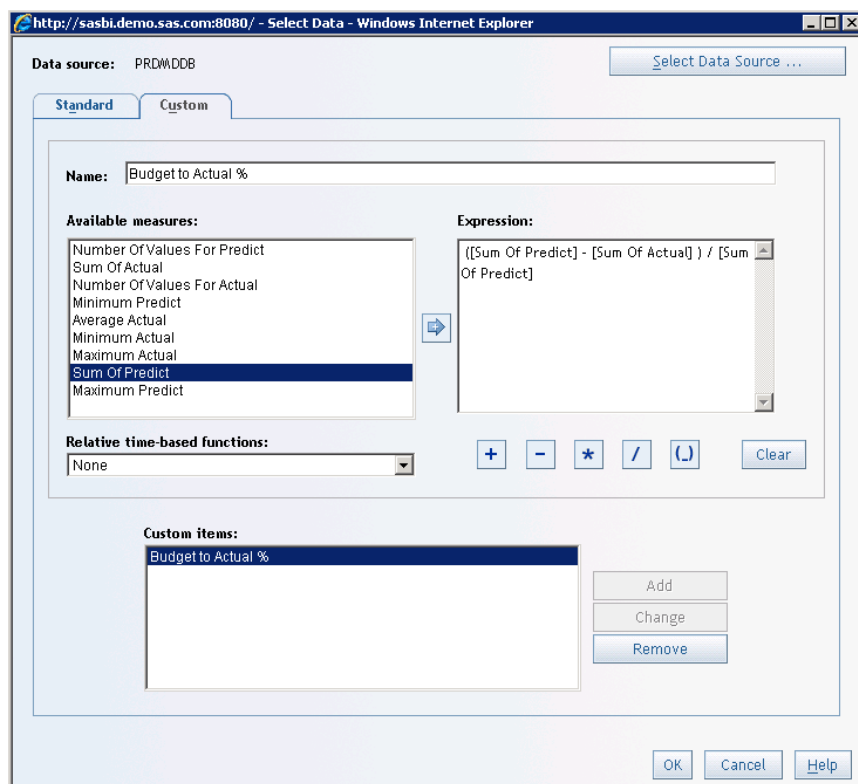


Figure 20: SAS Web Report Studio. Custom tab in Select Data interface allows define expressions for Custom Items

Here you can code basic manipulations with existing measures and use built-in Relative time-based functions.

Figure 20 shows the creation of our familiar [Budget to Actual %] Custom Item.

For a more sophisticated customization, you can use an Information Map that the report is based on and add Data Items on that level as described earlier. Also, CMs can be added on a cube level or created in a Global scope via other tools which makes them visible via SAS Information Map Studio and SAS Web Report Studio for further reporting.