Paper 330-2010

Enhancements for Managing SAS® OLAP Server in SAS® 9.2

Matthias Ender and Tatyana Petrova, SAS Institute Inc., Cary, NC

ABSTRACT

SAS 9.2 has enhancements to SAS OLAP Server that provide OLAP users and administrators more flexibility and control over their OLAP environment. With these enhancements, OLAP users can do the following:

- Distribute the workload across servers using load balancing
- Control sessions and cubes using PROC OLAPOPERATE and SAS® OLAP Server Monitor
- Secure cubes better

This paper is targeted to advanced SAS users and BI administrators. It highlights when and how these enhancements can be used to provide the most benefits to the users of SAS OLAP Server.

ENHANCEMENT 1: LOAD BALANCING

Load balancing for OLAP servers is a new feature in SAS 9.2.

The most important thing to know about load-balancing OLAP servers is that it makes sense only in multi-user situations. If you have a high query concurrency and reason to believe that the concurrency leads to lower performance for individual queries, then it is time to consider adding another OLAP server process.

The server instances in a load-balanced OLAP server cluster might be all on the same machine or on different machines.

Running multiple OLAP server processes on the same machine makes sense because of absolute process restrictions (2 GB on 32-bit systems) or pragmatic resource utilization limits (4 GB per CPU for a SAS OLAP Server process on any platform).

Running multiple OLAP server processes on different machines makes sense when you need to scale beyond the resources available on a single machine.

OLAP servers balance their load by user session. All query requests within a user session are executed by the same server instance. The first OLAP server that is started becomes the “master.” It receives all of the client requests and redirects them, as needed. If a new session is requested, the master forwards the request to the next OLAP server with the lowest session count.

All server instances access the same cubes. That means that a multiple-machine setup needs access to the same physical location.

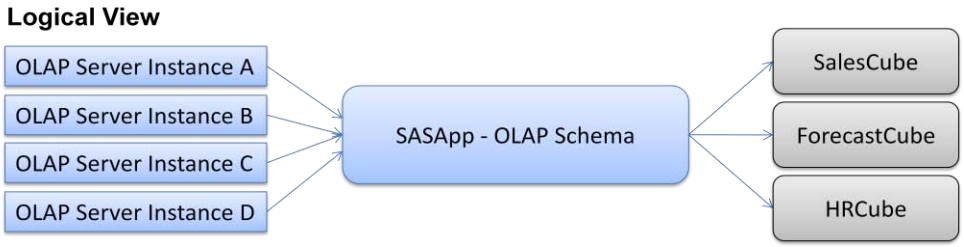

Figure 1. Logical View

The logical view shows the metadata registrations of the servers and cubes. All instances of a load-balanced OLAP server point to the same OLAP schema and use the same set of cubes.

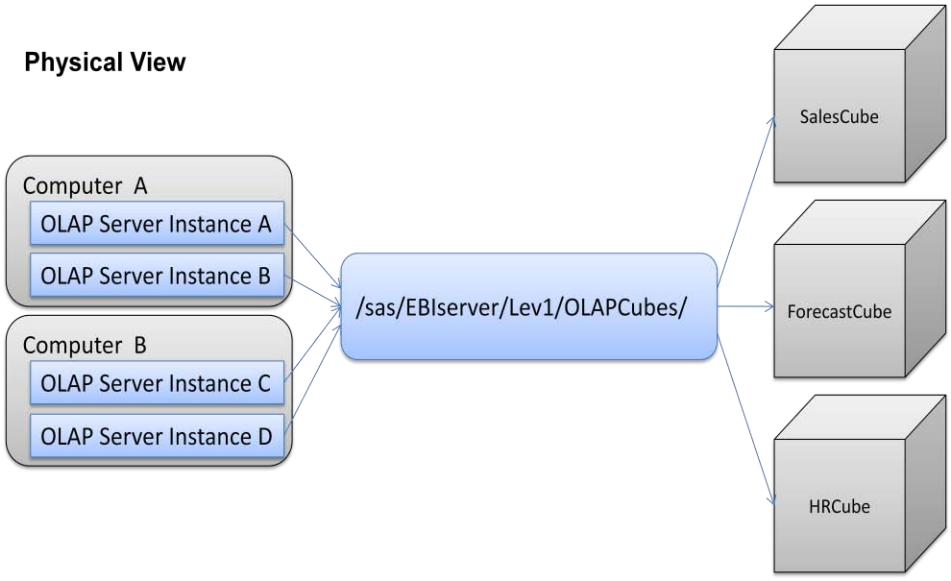

Figure 2. Physical View

The physical view shows the physical instances of the servers on different machines. It also shows the physical location of the cubes. All instances of a load-balanced OLAP server access the same physical cube location.

Load-Balanced Server Setup

To create load-balanced OLAP server instances, convert a regular OLAP server to a load-balanced OLAP server in SAS® Management Console:

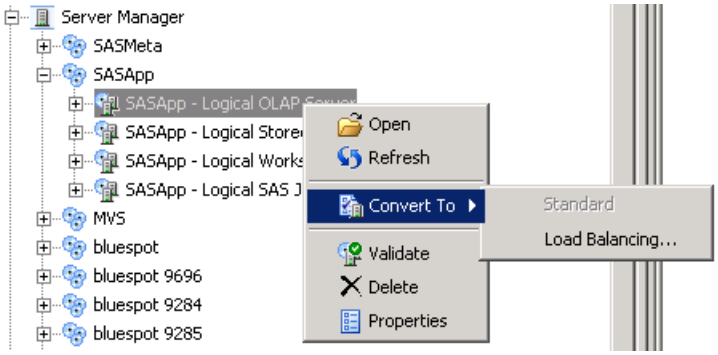

Figure 3. Converting a Regular OLAP Server to a Load-Balanced OLAP Server

Add as many instances of OLAP servers as needed. Set the server name and port of each server instance in the properties of each server's metadata definition.

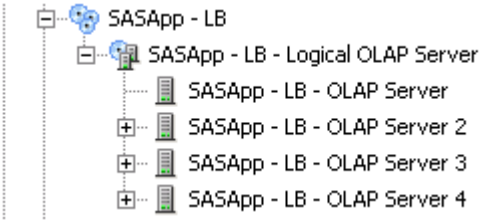

Figure 4. Example of Four Load-Balanced Servers

In addition, copy and modify the configuration directories and start-up scripts for each server instance.

For more details, see "Understanding Server Load Balancing" in the *SAS 9.2® Intelligence Platform: Application Server Administration Guide*.

The following graphs (Figures 5 through 8) show the effect of adding load-balanced OLAP servers on the multi-user scalability of a set of queries on a single server machine and on two server machines. A set of 11 representative queries was chosen and submitted continuously, adding up to 100 simultaneous user sessions. The graphs plot the average query response time and the relative response time changes of the queries.

Absolute performance numbers are subject to many factors, including the selection of cube and queries, hardware, and testing setup. These graphs show only the principle of load-balancing and scalability. They are not a performance reference. It would exceed the purpose and space of this paper to publish a complete performance test setup. One item to note—the tests to create this graph were run continuously with minimal "think time" between each test. That means that the number of concurrent virtual users shown would translate into a much larger number of actual users because real users who are driving interactive interfaces do not continuously submit queries.

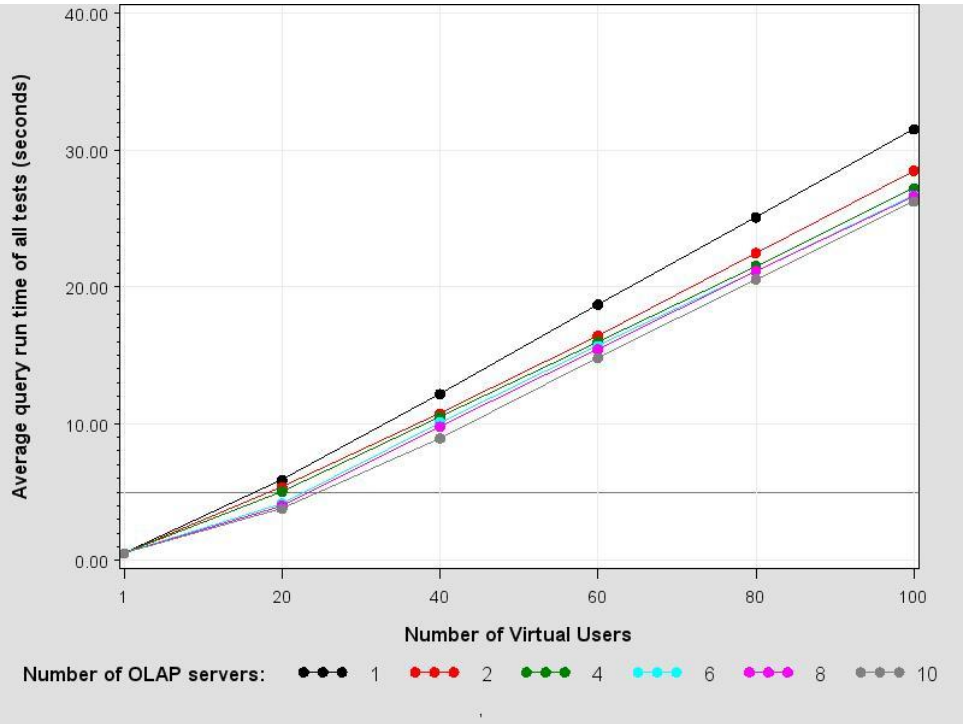

Figure 5. Average Query Run Time with Load-Balanced OLAP Servers on a Single Machine

One way to read Figure 5 is to assume that the maximum acceptable average response time is set at five seconds. With one server instance, you can accommodate up to about 16 users within that performance target. With four server instances, you can accommodate up to about 20 users.

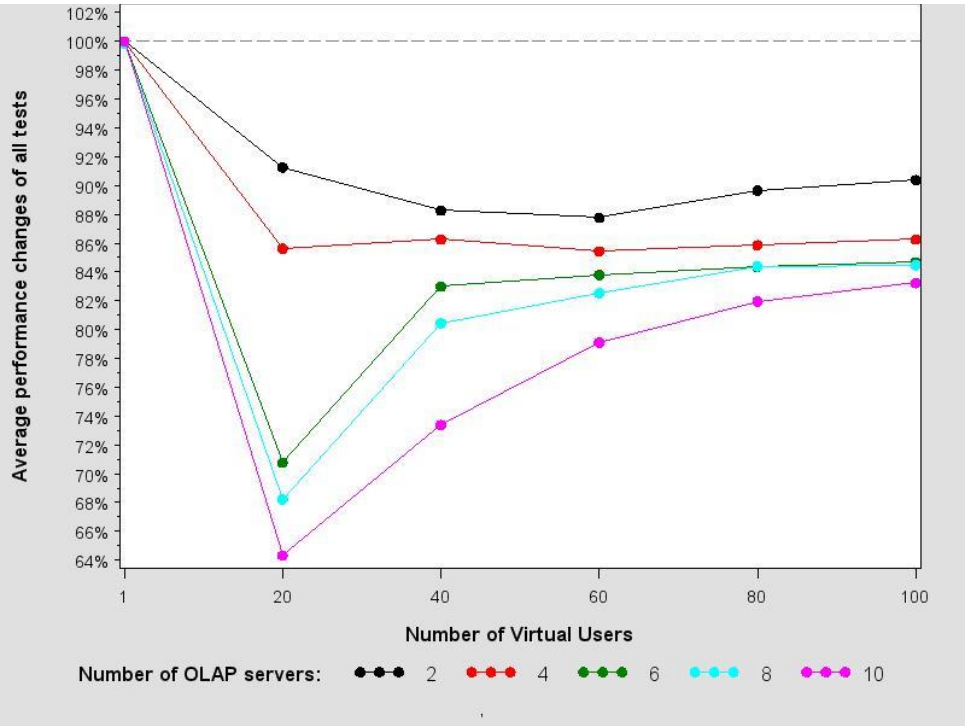

Figure 6. Average Query Run-Time Improvements with Load-Balanced OLAP Servers on a Single Machine

Figure 6 is another way of looking at the same data. It illustrates the performance improvement as a percentage of the single-server run time. Overall, performance improved 10% to 20%, with some spikes within the lower user concurrencies.

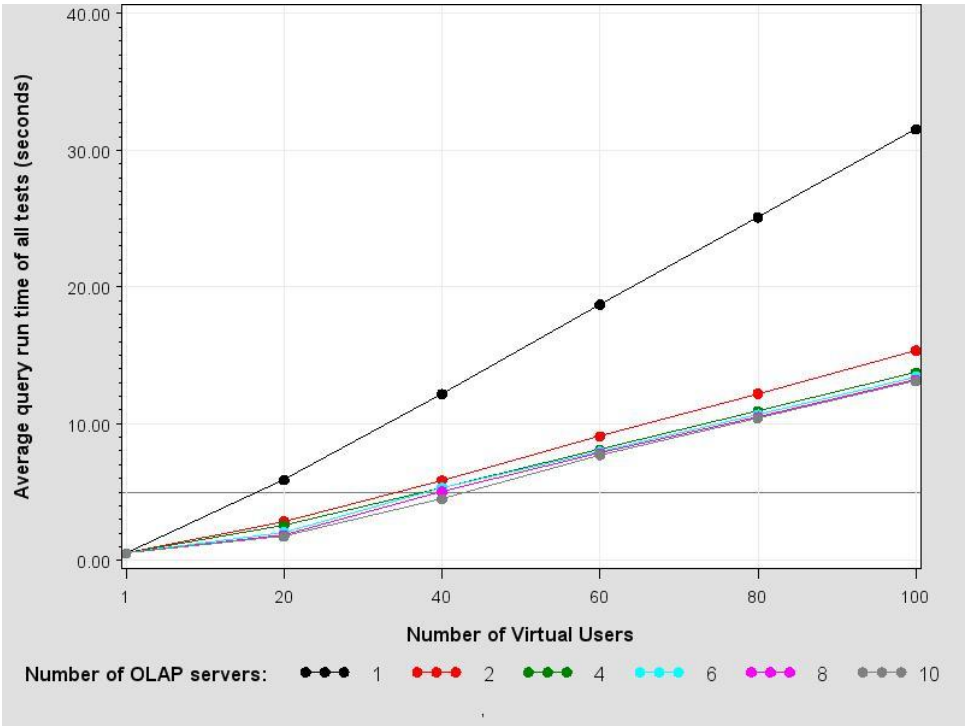

Figure 7. Average Query Run-Time Improvements with Load-Balanced OLAP Servers on Two Machines

As seen in Figure 7, when distributing the server instances over two machines, the performance gain is greater than when running multiple instances on a single machine.

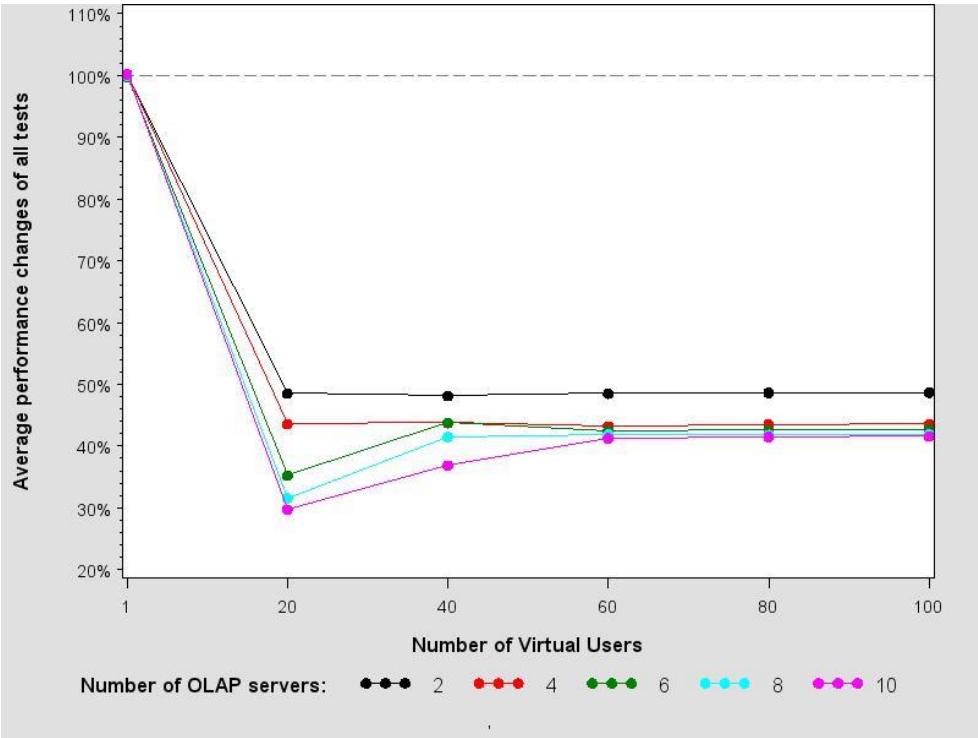

Figure 8. Average Query Run-Time Improvements with Load-Balanced OLAP Servers on Two Machines

Figure 8 shows more clearly that by adding a server instance on a second machine, the average query response time for any number of users is cut almost exactly in half. It shows that by adding further server instances to the two machines, an improvement of an additional 10% or so can be achieved.

ENHANCEMENT 2: PROC OLAPOPERATE AND SAS OLAP SERVER MONITOR

PROC OLAPOPERATE is a new procedure in SAS 9.2. SAS OLAP Server Monitor has been expanded in SAS 9.2 to include more functionality.

PURPOSE OF PROC OLAPOPERATE AND SAS OLAP SERVER MONITOR

PROC OLAPOPERATE and SAS OLAP Server Monitor help you control your OLAP servers. SAS OLAP Server Monitor is a SAS Management Console plug-in that gives you an interface to manage servers interactively in a familiar OLAP administrative environment. PROC OLAPOPERATE is a SAS procedure that enables you to automate your administrative steps.

WHAT YOU CAN DO WITH THESE TOOLS

In the following code, PROC OLAPOPERATE demonstrates this functionality. PROC OLAPOPERATE and SAS OLAP Server Monitor overlap in most of the operations.

```
PROC OLAPOPERATE [ <connection options> ];
    CONNECT <connection options>;
    DISCONNECT;
    LIST SESSIONS [CUBE=cube-name] [USER=user-id];
    LIST QUERIES [SESSION=owner];
    LIST ROWSETS [SESSION=owner];
    CLOSE SESSION id | _ALL_ ;
    CANCEL QUERY id | _ALL_ ;
    DISABLE CUBE cube-name | _ALL_ ;
    ENABLE CUBE cube-name | _ALL_ ;
    REFRESH CUBE cube-name | _ALL_ ;
    STOP SERVER;
RUN;
```

Functionality can be categorized into three groups of statements: connect to the server, list activities on the server, and manage the server.

1. Connect to the OLAP server. PROC OLAPOPERATE provides two ways to connect. The first way is to use connection options in PROC OLAPOPERATE:

```
PROC OLAPOPERATE HOST="localhost" USERID="SGFdemo"
     PW="logSGF1" PORT=5451 PROTOCOL=BRIDGE;
RUN;
```

   The second way is to use the CONNECT statement:

```
PROC OLAPOPERATE;
  CONNECT HOST="localhost" USERID="SGFdemo"
     PW="logSGF1n" PORT=5451 PROTOCOL=BRIDGE;
RUN;
```

   After you are connected, you can issue commands to this server. The CONNECT statement and other statements should be issued in the same procedure.

2. List sessions that are running by a session owner, by a cube, or all sessions by default. List queries that are running and list row sets that are returned by a session owner or all by default.

```
PROC OLAPOPERATE;
  CONNECT HOST="localhost" USERID="SGFdemo"
     PW="logSGF1n" PORT=5451 PROTOCOL=BRIDGE;
  LIST SESSIONS;
  LIST QUERIES;
  LIST ROWSETS;
RUN;
```

   For this example, one session was running on the OLAP server and one query was submitted to the ORIONCUBE cube. Here is the output from the procedure:

```
NOTE: The CONNECT command completed.

Session ID: 2E6A7F4A-6B2C-4995-B694-55133F9EDAE0
Session owner: SGFdemo@d18055
Session name:
Session description:
Session seconds inactive = 54 and open results = 5
Cube ORIONCUBE is active in this session.
NOTE: The LIST SESSIONS command completed.

Query Statement "FROM [ORIONCUBE] SELECT  {[CUSTOMERS].[ALL
CUSTOMERS].CHILDREN} ON COLUMNS,
{[TIME].[YMD].[ALL YMD].[2000]} ON ROWS "
Query ID: 9232670F-BD64-412C-B933-FA81E2FDB286
Query type is Multidimensional
Query size in bytes =  686896
Total cells = 3 and cells read = 3
The time of last data set access is 24Jan2010:12:36:46
NOTE: The LIST QUERIES command completed.

Rowset name "CUBESET"
Rowset ID: 4A0C6D56-850A-48AA-A850-68EC9BB9A484
The time of last rowset access is 24Jan2010:12:36:31

Rowset name "DIMENSIONSET"
Rowset ID: E16BB3C5-BA5F-4B38-8DC2-642E10CFA5D5
The time of last rowset access is 24Jan2010:12:36:32
Cube ORIONCUBE is being accessed by this rowset.

Rowset name "HIERARCHYSET"
Rowset ID: 8584B146-043E-48A6-813B-A15E782F3676
```

```
The time of last rowset access is 24Jan2010:12:36:38
Cube ORIONCUBE is being accessed by this rowset.

Rowset name "LEVELSET"
Rowset ID: 2BD30153-5924-4DC2-8F78-70EC0B1E4715
The time of last rowset access is 24Jan2010:12:36:39
Cube ORIONCUBE is being accessed by this rowset.
NOTE: The LIST ROWSETS command completed.

NOTE: PROCEDURE OLAPOPERATE used (Total process time):
      real time           0.18 seconds
      cpu time            0.01 seconds
```

3. The rest of the statements enable you to manage the server using general options and based on the output returned by the LIST commands.

   When you know who is connected to the server and what is running on the server, you can cancel queries that have been running too long, close inactive users sessions, disable or enable cubes for maintenance scheduled, stop the server, and so on.

   For example, using the output returned by the LIST commands, you can issue the following commands:

```
PROC OLAPOPERATE;
  CONNECT HOST="localhost" USERID="SGFdemo"
     PW="logSGF1n" PORT=5451 PROTOCOL=BRIDGE;
  CANCEL QUERY "9232670F-BD64-412C-B933-FA81E2FDB286";
  DISABLE CUBE "ORIONCUBE";
  CLOSE SESSION "2E6A7F4A-6B2C-4995-B694-55133F9EDAE0";
  STOP SERVER;
RUN;
```

If providing session and query IDs, and writing code are not your idea of fascination, the same functionality can be performed using SAS OLAP Server Monitor.

Connect to the OLAP server.

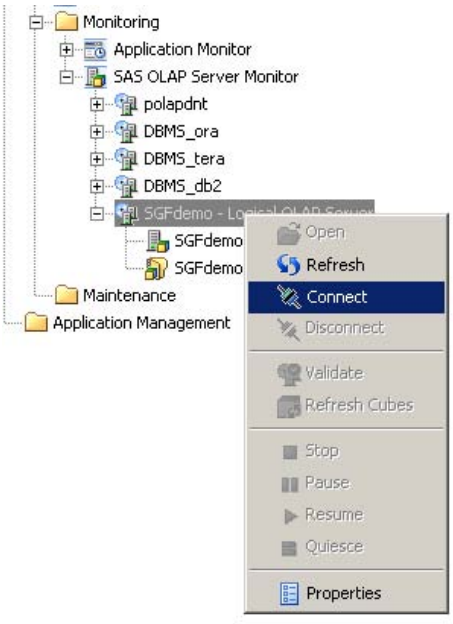

Figure 9. The Connect Option in SAS OLAP Server Monitor

If connection was successful, additional operations on the server to which you are connected are available:

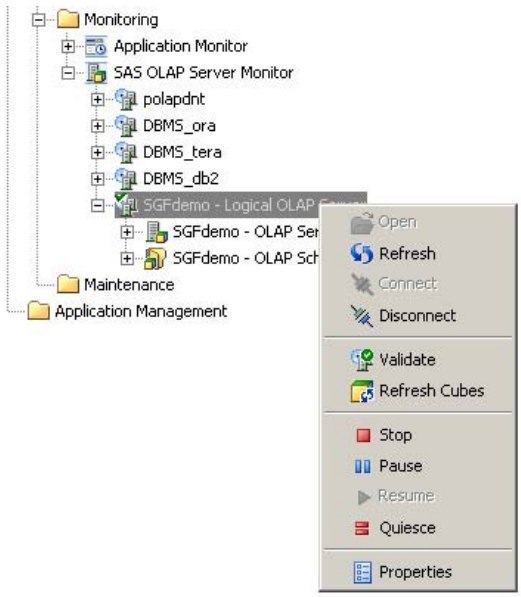

Figure 10. Additional Options in SAS OLAP Server Monitor

You can monitor which users are connected to and what queries are running on the server. More detailed information about users and queries is available in the right pane of SAS Management Console. You can cancel queries and close sessions on this server:

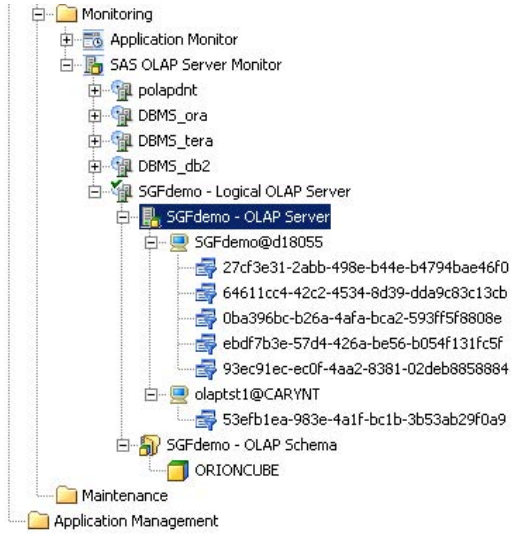

Figure 11. Active Sessions Displayed in SAS OLAP Server Monitor

USE CASE SCENARIO

Consider this task. As part of a managed cube update, old and updated cubes should be swapped. To swap cubes, you need the updated cube ready. Then, you need to switch the online version of the cube from the old version to the new version, while minimizing offline time and any impact on users.

Depending on company policies and user requirements, there are several ways to update a cube and swap cubes. Here are possible steps to swap cubes:

- List queries using PROC OLAPOPERATE or SAS OLAP Server Monitor. Cancel queries that are running against the cube. Pause the OLAP server (in SAS 9.2, this feature is available only in SAS OLAP Server Monitor) so that no new queries get submitted.
- Disable the old version of ORIONCUBE with PROC OLAPOPERATE or SAS OLAP Server Monitor.

- Rename the old version of ORIONCUBE to have a different name with PROC OLAP or SAS OLAP Cube Studio.
- Rename updated version of the cube to ORIONCUBE.
- Enable the ORIONCUBE cube with PROC OLAPOPERATE or SAS OLAP Server Monitor.
- If the server has been paused, resume it (in SAS 9.2, this feature is available only in SAS OLAP Server Monitor).

ENHANCEMENT 3: SECURING CUBES BETTER

There are several SAS 9.2 tool enhancements for securing OLAP cubes. For example, changes were made to the rules for security inheritance in the OLAP data structure. There are enhancements to a member-level security application GUI. An option was added to cube build syntax to exclude aggregated values (security totals) for members that users have no access permissions to. Functionality was added to the OLAP server to honor the security that was applied on relational data structures for drill-through querying. A wizard was added to SAS OLAP Cube Studio for security application automation.

SECURITY INHERITANCE

In SAS 9.1.3, an OLAP schema was a common level on which OLAP security could be defined in the metadata structure. (For example, a user could have ReadMetadata permission defined for an OLAP schema and, therefore, for all cubes that belong to this OLAP schema). In SAS 9.2, permission is determined by the parent folder. All OLAP metadata objects (for example, cubes, dimensions, levels, and measures) that are placed in a metadata folder inherit the security settings from the folder. (See Figure 12.)

This security inheritance is important to remember when defining the structure of your OLAP metadata or when moving OLAP metadata objects within this structure. It is especially important to remember when you use the easy drag-and-drop functionality in the new folder tree in SAS Management Console in 9.2 and in SAS OLAP Cube Studio 4.2.

Because the OLAP schema has lost its distinction as a security designation point, migration tools do not move security that is set on a schema when they migrate metadata from 9.1.3 to 9.2. If administrative users controlled OLAP security using OLAP schemas, they now need to reset security using the new folder tree.

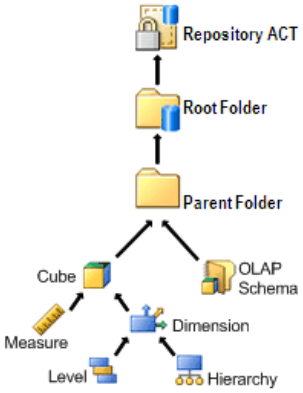

Figure 12. Security Inheritance in the OLAP Data Structure

You can overwrite inherited security settings on every level below the parent folder (for example, cubes, dimensions, levels, and so on). SAS 9.2 OLAP enhancements include a more robust interface for creating filters on a dimension level. This approach is known as member-level security.

MEMBER-LEVEL SECURITY APPLICATION GUI

Member-level security is not new in SAS 9.2. What *is* new are the GUI improvements to walk through the setup process. A new set of dialog boxes supports the building of permission conditions in SAS Management Console and SAS OLAP Cube Studio.

To get to the GUI in SAS Management Console, perform the following steps:

- Open the Authorization Manager.
- Select a cube.

- Right-click on a dimension within this cube, and then select **Properties**.

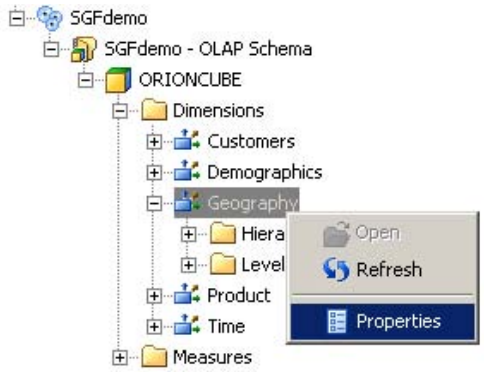

Figure 13. Getting to the Member-Level Security Application in SAS Management Console

- Click on the **Authorization** tab in the properties dialog box.
- Select or add the user or group who will have the specified filter.
- Grant an explicit Read permission to the user or group.
- Overwrite the inherited Read permission for this user or group to enable the Add Authorization dialog box. To do this, click on a Read permission for the user or group, and make sure that it is not dimmed. An **Add Authorization** button appears.
- Click **Add Authorization**.

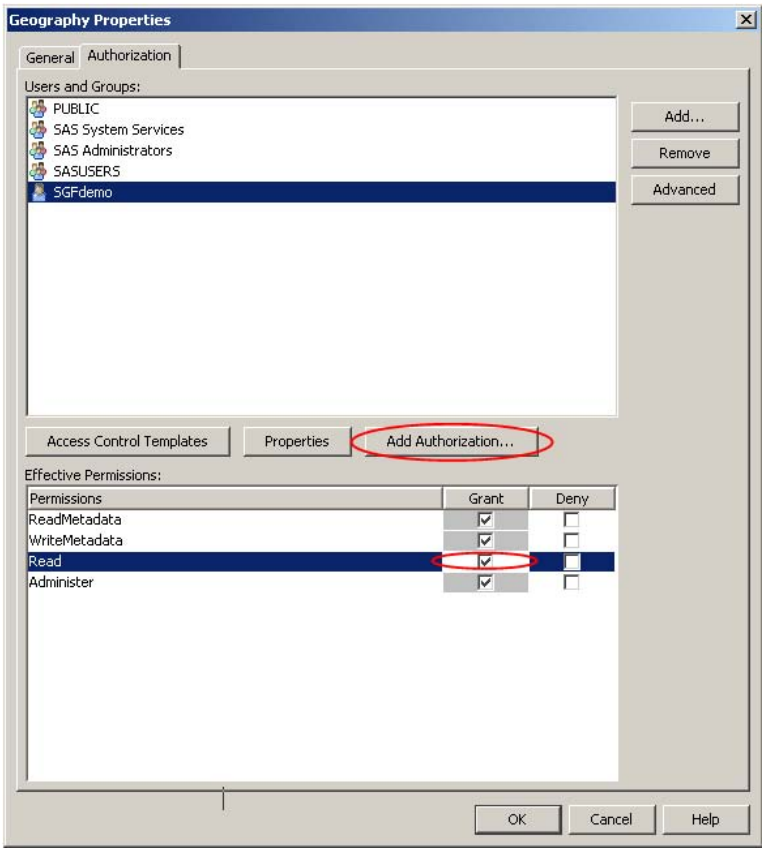

Figure 14. Getting to the Member-Level Security Application in SAS Management Console

Remember that permission conditions can be specified only on dimension objects in the metadata object structure in SAS Management Console or in SAS OLAP Cube Studio. You cannot enable the Add Authorization dialog box if you initiate the properties dialog box in a cube level below the dimension level. In the **Add Authorization** dialog box itself, you can work with any level of the dimension.

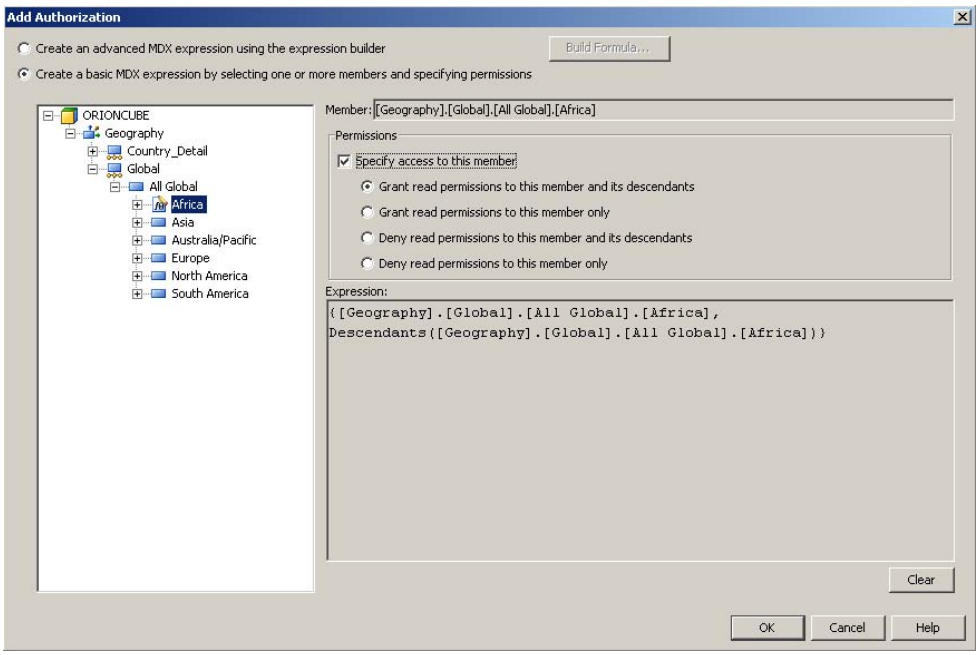

Figure 15. Member-Level Security Application

To build a basic MDX expression for the permission conditions specification, you can use the interactive field in the Add Authorization dialog box. In addition, you can use the metadata object structure in the left pane of the Add Authorization dialog box to select cube members to which you want to grant or deny access.

To build more advanced code, you can initiate the Build Formula dialog box by clicking the **Build Formula** button. The Build Formula dialog box provides a field to build a more complicated MDX expression, and provides access to MDX syntax. In the Build Formula dialog box, you can use the built-in syntax provider or you can type or copy code. For this example, the expression was built by adding `[Geography].[Global].[All Global]` to an MDX expression that was generated in a previous step.

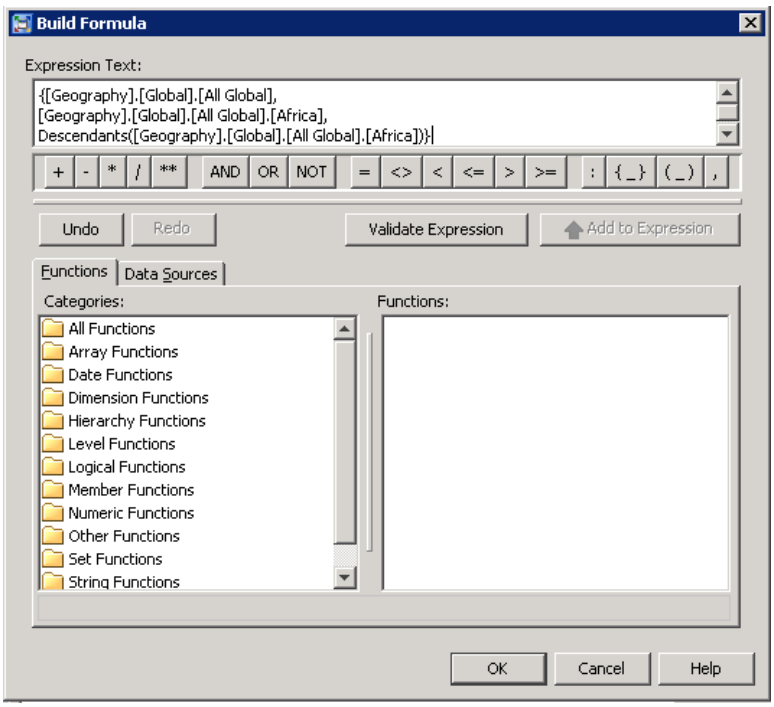

Figure 16. Build Formula Dialog Box

SECURITY TOTALS AND NEW SECURITY_SUBSET OPTION IN PROC OLAP

Before SAS 9.2, permission conditions did not affect the rolled-up values for measures at query time. In SAS 9.2, you can set a cube option that controls whether your aggregated values honor permission conditions at query time.

Consider the following example in which a permission condition is created for the user for dimension `[Geography]`:

```
{[Geography].[Global].[All Global],
[Geography].[Global].[All Global].[Africa],
Descendants([Geography].[Global].[All Global].[Africa])}
```

Before SAS 9.2, aggregated values for **All Global** included values for members to which the user did not have access:

			Measures	Sum of Total_Cost
All Global	*Continent*	*Country*		
All Global				$62,468,540.83
	Africa			$76,387.35
		Benin		$368.35
		Ivory Coast		$1,661.20
		Morocco		$1,669.70
	Africa	Mozambique		$618.40
		Nigeria		$114.15
All Global		Senegal		$1,444.70
		Tunisia		$1,233.85
		South Africa		$69,277.00

Figure 17. Aggregated Values Include Values for Restricted Members

In SAS 9.2, you can control whether to exclude restricted member values from the aggregated values.

			Measures	Sum of Total_Cost
All Global	Continent	Country		
All Global				$76,387.35
All Global	**Africa**			$76,387.35
	Africa	Benin		$368.35
		Ivory Coast		$1,661.20
		Morocco		$1,669.70
		Mozambique		$618.40
		Nigeria		$114.15
		Senegal		$1,444.70
		Tunisia		$1,233.85
		South Africa		$69,277.00

Figure 18. Aggregated Values Do Not Include Values for Restricted Members

To honor a session's permission conditions, set the new option in the PROC OLAP statement `SECURITY_SUBSET = YES:`

```
PROC OLAP CUBE=ORIONCUBE SECURITY_SUBSET= YES;
METASVR HOST="localhost"
        PORT=5451
        PROTOCOL=BRIDGE
        USERID="SGFdemo"
        PW="logSGF1n"
        REPOSITORY=Foundation
        OLAP_SCHEMA="SGFdemo - OLAP Schema";
RUN;
```

With PROC OLAP, you can set this option at cube build time or change it on an existing cube without rebuilding the cube.

The same function can be accomplished using the SAS OLAP Cube Studio interface:

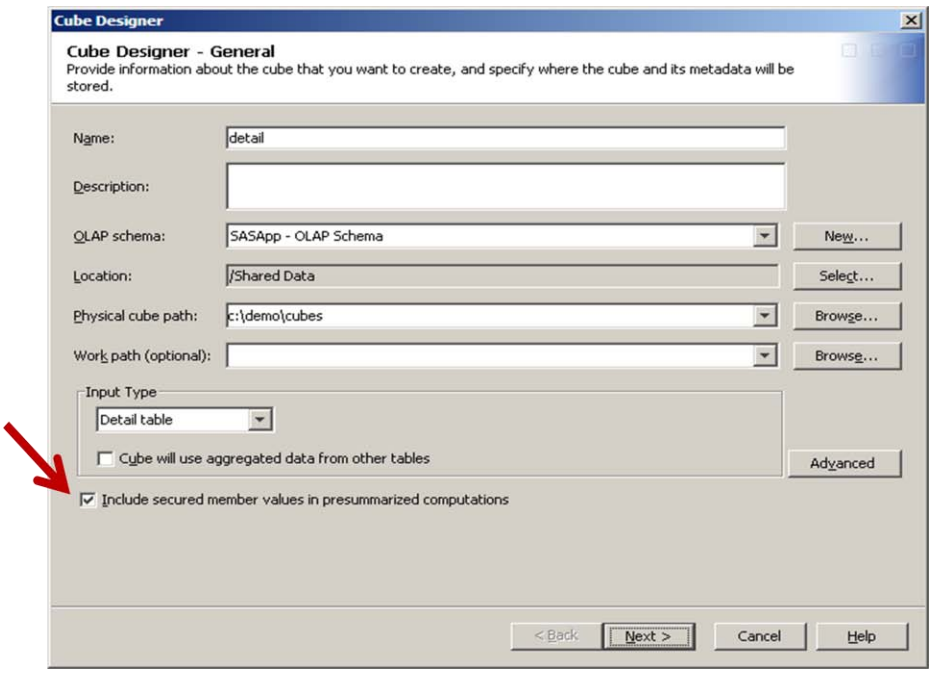

Figure 19. SAS OLAP Cube Studio Interface

In the Cube Designer, the check box is selected by default, which means that the SECURITY_SUBSET option is set to NO. If you leave the check box selected, permission conditions are ignored. Ignoring permission conditions by default ensures consistency in reports created with SAS 9.2 and previous releases. To honor the permission conditions of a cube, make sure this check box is not selected.

SECURITY ON OLAP CUBES FOR DRILL-THROUGH QUERYING

Starting with the second maintenance release for SAS 9.2, security on OLAP drill-through tables is honored. In other words, the credentials of the user that sees the OLAP report are passed down to the drill-through table. Dropping columns is straightforward—you deny ReadMetadata permission on the column in the table registration in SAS Management Console. Row-level security requires an information map. You define the filters in the information map, and then use the information map as a drill-through table.

Dropping Columns

Consider this example. A user is not allowed to drill to the level [EMPLOYEE_NAME], which is the lowest level of the Organization dimension in the cube. You want to deny access to the corresponding column in the drill-through table.

To set the permissions, select the drill-through table's properties dialog box (for example, from the Library Manager in SAS Management Console). On the **Columns** tab, you can access the column's Authorization dialog box. Be sure to deny ReadMetadata permission for the column.

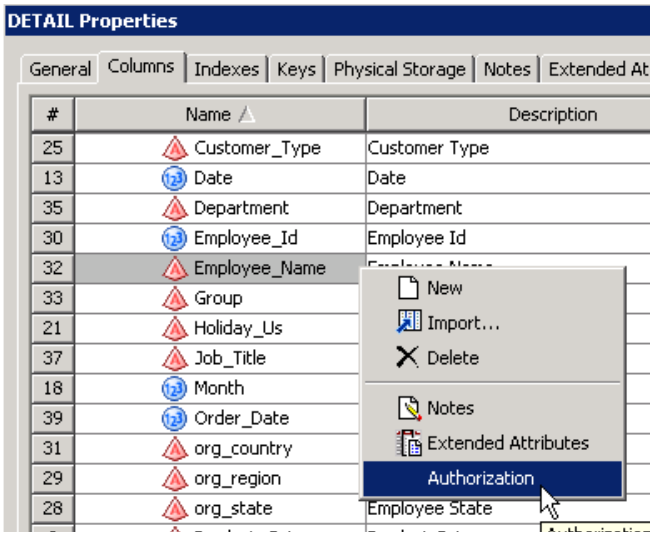

Figure 20. Accessing the Authorization Dialog Box for a Column

Be sure to deny access only to columns that correspond to levels in the cube that the user is not allowed to see. Otherwise, the drill-through queries might fail.

Dropping Rows

If you want row-level security on the drill-through table, you must use a SAS library engine that accesses data sources that provide row-level security.

You can use SAS Information Map Studio and access information maps as SAS tables using the SAS Information Maps LIBNAME Engine. If you define your cube so that the drill-through table is in a SAS Information Maps library, then the drill-through query will pick up any row-level security specified in the information map.

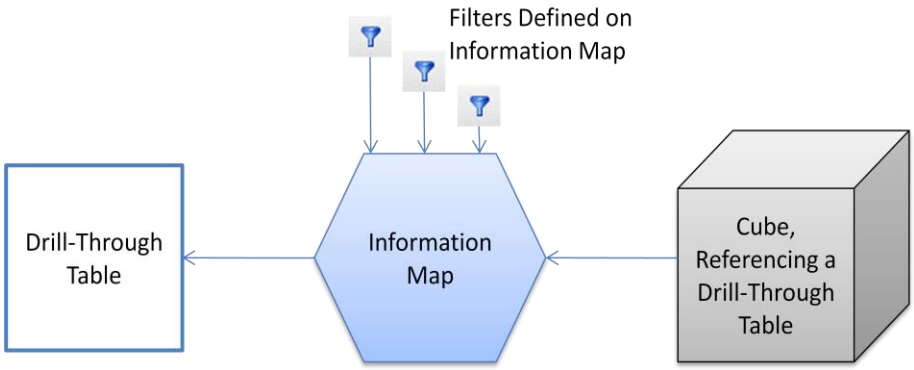

Figure 21. Implementing Row-Level Security for Drill-Through Tables Using SAS Information Map Studio

In SAS Information Map Studio, in your information map, define one or more filters for each user or group.

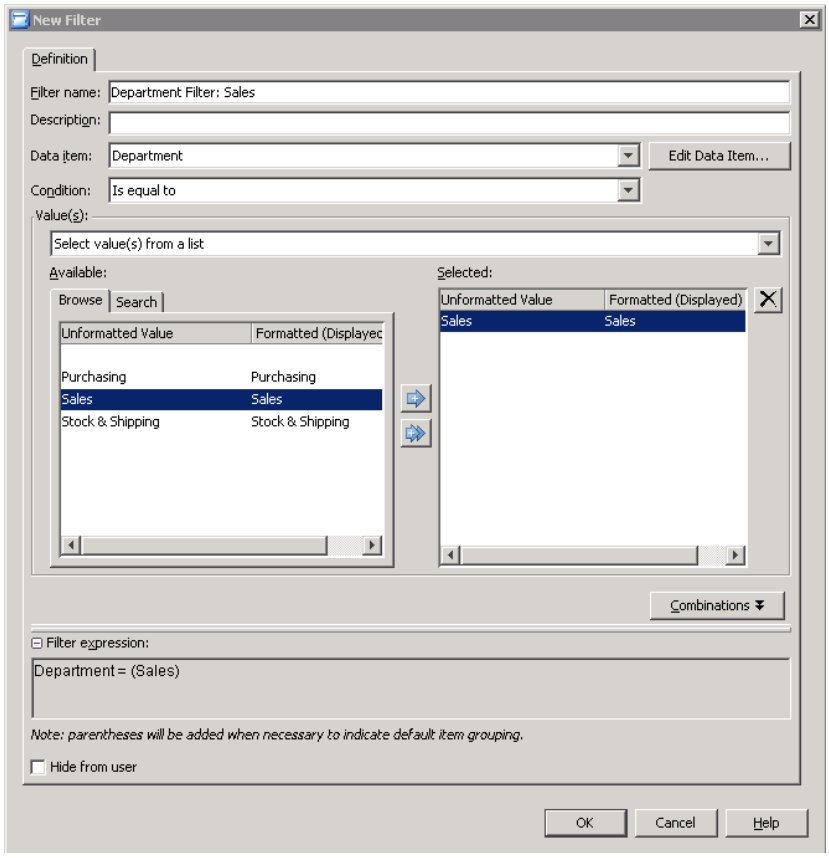

Figure 22. SAS Information Map Studio—Adding a New Filter

In the Authorization dialog box, assign filters to users or groups.

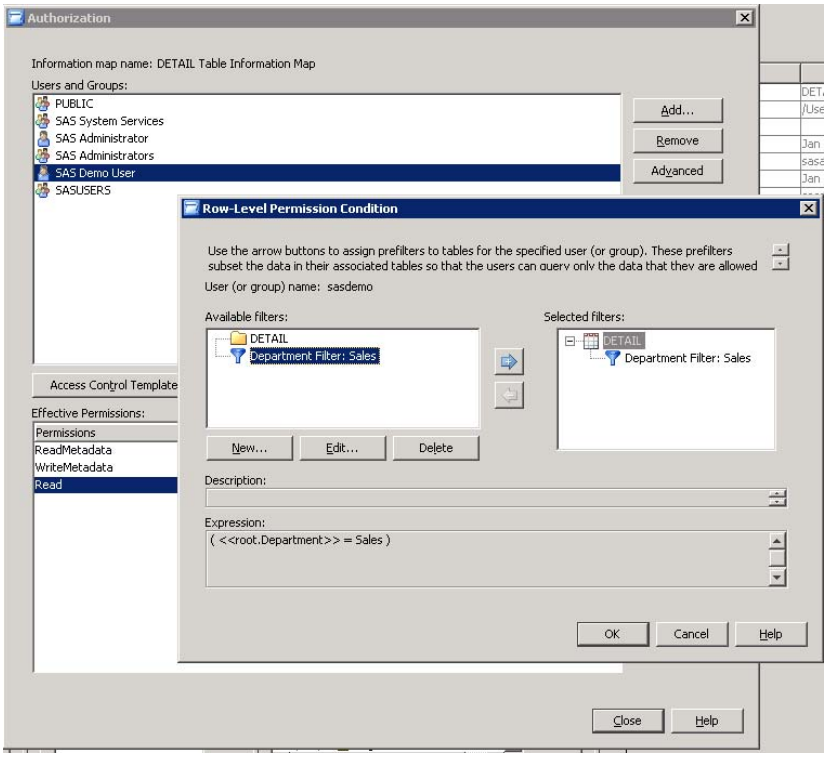

Figure 23. SAS Information Map Studio—Assign Filter to User or Group

Use SAS/ACCESS library engines to other relational data structures to pick up row-level security defined in external relational databases.

SECURITY APPLICATION AUTOMATION WIZARD

If you need to maintain many permissions conditions, use the GUI- supported batch security wizard that was introduced in second maintenance release for SAS OLAP Cube Studio 4.2. It enables you to import cube permissions from a SAS table. You can create and modify the SAS table using your own custom program or using the provided table editor.

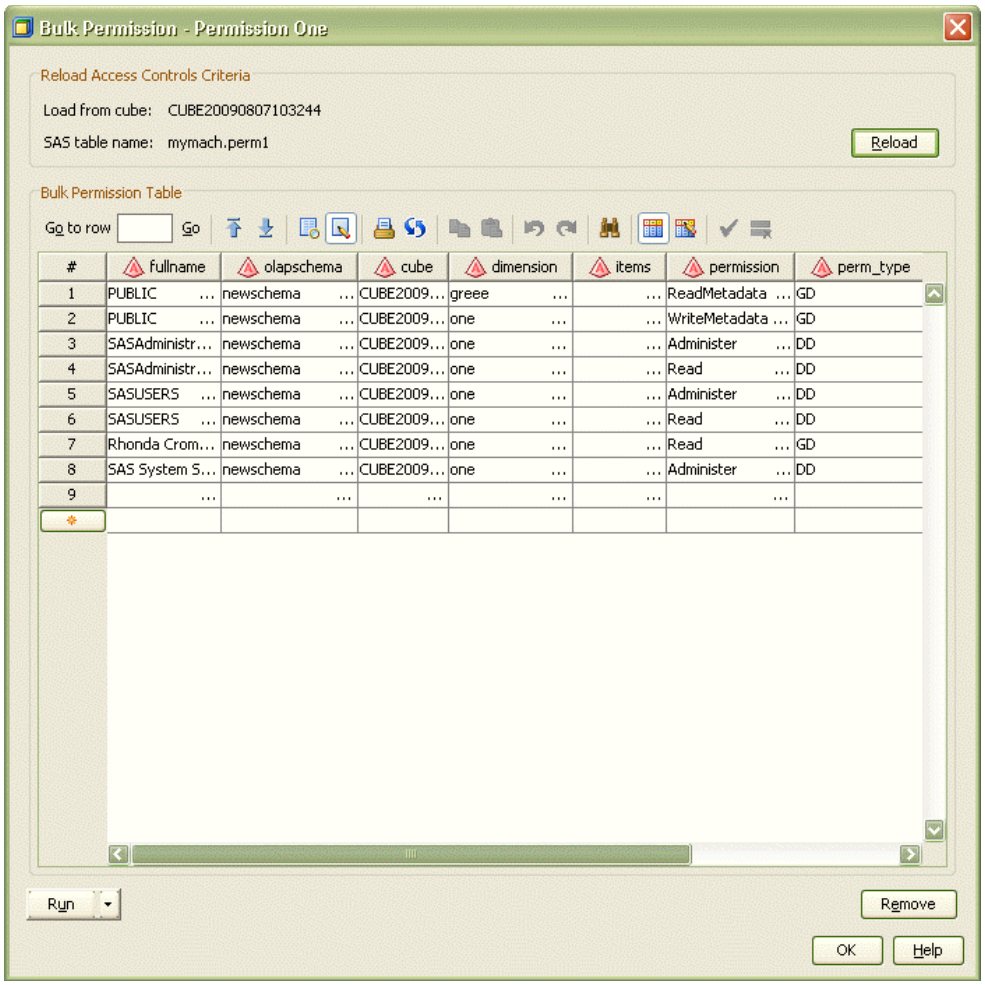

Figure 24. Security Application Automation Wizard in SAS OLAP Cube Studio—Table Editor

USE CASE SCENARIO

Consider this task. You have an experimental cube named ORIONCUBE_test with a set of user security restrictions applied. You want to apply these restrictions with a few updates to the working version of a cube named ORIONCUBE.

There are several ways to accomplish this task. For example:

- Open SAS OLAP Cube Studio and select **Tools**->**Manage Permission Tables**. The Manage Permission Tables dialog box appears and lists all permission tables registered on the SAS Metadata Server.
- Create a new permission table based on the settings for the ORIONCUBE_test cube. To do this, click **Add**.
- In the Add Permission Table dialog box, specify the name of the table (as it will appear in the metadata), the library in which the table will reside, and the SAS table name (data set name). In the next dialog box, select the cube (in this case, ORIONCUBE_test) from which you want to load security data. When you click **Finish**, the data set is created that will hold all of the data about security settings on that cube.

  If any errors occur during the creation process (for example, the servers are not running, permission is lacking for the user who initiated the table creation, or so on), you get an error message. You can save the data set creation code (the security settings *retrieval* program) as a separate file so that you can debug or update it later.

  The user who works with permission tables needs to have WriteMetadata permission to the SAS Metadata Server, the SAS Workspace Server, and the library that holds permission tables. The user needs WriteMemberMetadata permission to the folder that contains that library.

  Permission tables are SAS data sets. After they are created, they appear as registered tables in the library you specified. You can view and edit them as you would a SAS table, or process them using a SAS DATA step.

- After your table is successfully created, return to the Manage Permission Tables dialog box. You can modify security values using the table editor (see Figure 24) or using other methods. Apply the security settings from your permission table to a new cube or to the same cube from which the data was loaded. (Be sure to update the cube name in your table if you want to apply it to a new cube.)

  In this example, the ORIONCUBE_test cube needs to be updated to ORIONCUBE. If you use the table editor to update the cube, remember that each row change gets committed to the table when you leave the table row.

  When you make changes to the table, changes do not get automatically applied to the cube settings. To apply the changes to the cube, click **Run** in the table editor or **Apply** in the Manage Permission Tables dialog box. You can export code (the security settings *application* program) to update or automate it later.

  If any errors occur during the application process, you get an error message with access to the log. The log indicates which permission rows in the table applied successfully. It also provides details for those permission rows that failed.

- If you want to regenerate the table using settings already set on a cube, click **Reload** in the table editor. This will overwrite changes you have added to the table, but have not yet applied to the cube.

Here is the flow for processes in batch security application automation:

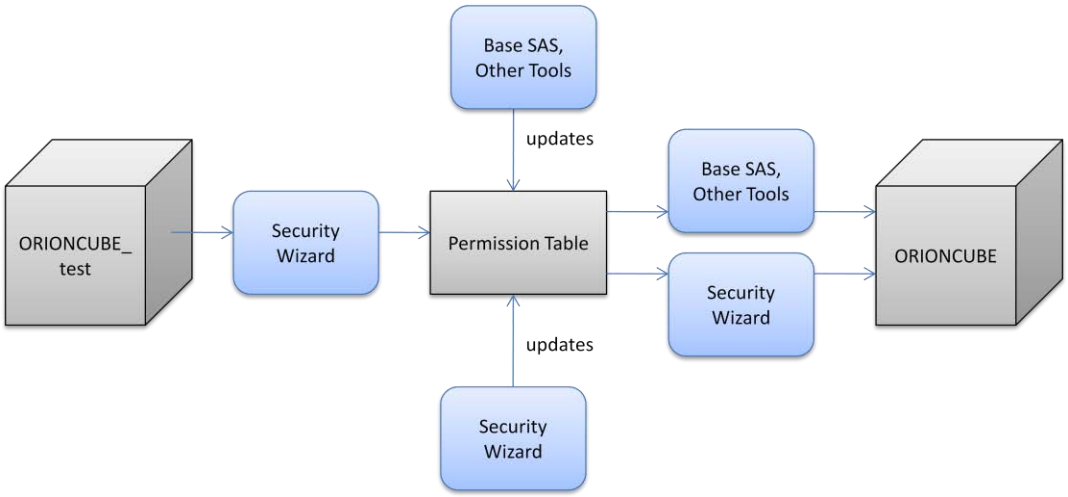

Figure 25. Batch Security Application Automation Flow

For more information, see “Modifying and Maintaining Cubes” in the *SAS OLAP Server: User’s Guide*.

CONCLUSION

All of the administrative tools presented in this paper can be used by the server administrator and the applications developer.

Load-balancing can be used for resource management (as a server administrator task) and end-user performance control. For OLAP applications, performance is a functional requirement. Load-balancing is an additional tool in the set of performance tools including aggregation tuning, hierarchy design, and report design.

Access control is often a task for the server administrator. However, it is also a productivity tool for the applications developer and enables user-based dynamic report generation.

Server monitoring and session control are technologies that help server administrators and applications developers with testing, debugging, tuning, and maintaining OLAP applications.

REFERENCES

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

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

SAS Institute Inc. 2009. *SAS 9.2 Intelligence Platform: Security Administration Guide*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2009. *SAS 9.2 Intelligence Platform: Application Server Administration Guide*. Cary, NC: SAS Institute Inc.

RECOMMENDED READING

SAS Institute Inc. 2009. SAS Institute white paper. "SAS 9.2 Software Key Highlights." www.sas.com/whitepapers.

CONTACT INFORMATION

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

Matthias Ender
SAS Institute Inc.
E-mail: Matthias.Ender@sas.com

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