Paper 045-2011

SAS® OLAP Cubes Go Green: Reusable Shared Dimensions in SAS® 9.3

Jana Van Wyk; Ann Weinberger, SAS Institute Inc., Cary, NC, USA

ABSTRACT

Do you find yourself needing to recreate the same dimension in multiple SAS OLAP cubes? SAS OLAP cube administrators and designers will be glad to know that, in SAS 9.3, you can create a dimension to be shared among multiple OLAP cubes. Re-using a shared dimension in multiple OLAP cubes helps ensure consistency and is also good for the environment, since it reduces the amount of metadata and member data stored.

This paper includes an introduction to this new feature and shows how to integrate it into your current OLAP environment. A case study is presented that starts with an existing cube and modifies it to use shared dimensions. The case study includes the application of authorization and also the promotion of shared dimensions and cubes. Best practices for maintenance of shared dimensions and cubes are also discussed.

INTRODUCTION

SAS OLAP cubes organize data in a hierarchical arrangement, according to dimensions and measures. Dimensions group the data along natural categories and consist of one or more levels. For example, a time dimension can include levels such as years, months, and days. Often the dimension data is shared across many different cubes. With previous releases of the SAS® OLAP Server, a dimension would be defined within the context of each cube that requires that dimension. As updates to the dimension definition and data are made, each cube using the dimension data would need to be updated separately. This approach created redundant metadata and member information while increasing the time required to maintain the dimensions.

In SAS 9.3, the SAS OLAP Server provides the ability to create shared dimensions. A SAS shared dimension provides a common dimension definition that is created and updated in one place and is automatically reflected across all cubes that use the dimension. A shared dimension enables the OLAP administrator to define the dimension once, rather than redefine the dimension for every cube that uses it. This saves processing time to build the dimension and disk space to store the member information as well as providing a consistent definition of the dimension across the enterprise. Advantages of using shared dimensions include the following:

- reduced storage of metadata
- reduced storage of member data
- consistent member information across cubes
- easier maintenance of the dimension metadata and member information

SHARED DIMENSIONS – THE BASICS

Like a private dimension in a cube, the definition of a shared dimension includes the levels, hierarchies, and member properties known to the dimension as well as attributes like options to describe ragged and unbalanced data. Shared dimensions are grouped together in an OLAP schema and are also contained in a folder in the SAS Folders tree. The data source for the shared dimension is a single dimension table (or a view of multiple tables). Thus, a shared dimension may be referenced only by a cube that is loaded from a star schema. The dimension table and dimension key are specified when the dimension definition is created, while the key in the fact table is specified when the dimension is used in a cube.

Unlike a private dimension in a cube, the definition of the shared dimension must also include a path where the physical files for the dimension will be stored. When the shared dimension is built, all physical files are built and stored in this location. Cubes that use the shared dimension will reference the files directly, both to build the cube and during query processing.

Updates to a shared dimension fall into two categories: adding new members within the existing structure and changing the existing structure (e.g. adding or dropping levels and/or hierarchies). The first type of update, adding new members, can be done while all cubes that are using the dimension remain online. The new members are added to the physical files for the shared dimension. The cubes using the shared dimension do not need to be rebuilt in order to use the new members. The new members will be returned, if appropriate, with the next query that opens the cube. However, there will be no fact data for these members until the cube has been updated.

The second type of update, changing the existing structure, is a more complex operation and requires a rebuild of the shared dimension physical files. Since the physical files of the shared dimension are referenced by all the cubes that

use the shared dimension, these cubes should be disabled and have their physical files deleted before the change to the shared dimension is made. Once the dimension has been rebuilt, the cubes can be rebuilt and enabled.

Access to the members of a shared dimension can be limited by setting permission conditions on the shared dimension. The permission condition is an MDX expression that filters the members of the shared dimension. The condition is defined for a shared dimension just as you would define a permission condition for a private dimension in a cube. However, in the case of a shared dimension, the permission condition applies to all cubes that use the shared dimension.

Shared dimensions are built using PROC OLAP code or a new wizard in SAS® OLAP Cube Studio, the Shared Dimension Designer. To include a shared dimension in a cube, PROC OLAP provides additional statements and keywords, and SAS OLAP Cube Studio provides additional prompts in the Cube Designer. SAS OLAP Cube Studio also provides tools to manage updating shared dimensions and the cubes that use the shared dimensions. These tools allow you to incrementally add member data to a shared dimension, edit the existing structure of a shared dimension, or rebuild a shared dimension. The Authorization tab of the property sheet for a shared dimension in SAS OLAP Cube Studio provides the ability to specify permission conditions for the shared dimension.

THE ORION CUBE

For our case study, the Orion STAR cube will be used. The following diagram shows the dimensional data model for this cube. The Orion STAR cube consists of a single fact table, Order_fact, and five dimension tables: Customer_Dim, Organization_Dim, Geography_Dim, Product_Dim, and Time_Dim. The tables are all contained in the library ORDATA. To simplify the code for our example, we will look at creating the Orion STAR cube with only two dimensions: Customer from the Customer_Dim table and Time from the Time_Dim table. We will show the steps to create and maintain the cube with all private dimensions and then show the steps to make the Customer dimension a shared dimension that is used along with the Time dimension in the cube.

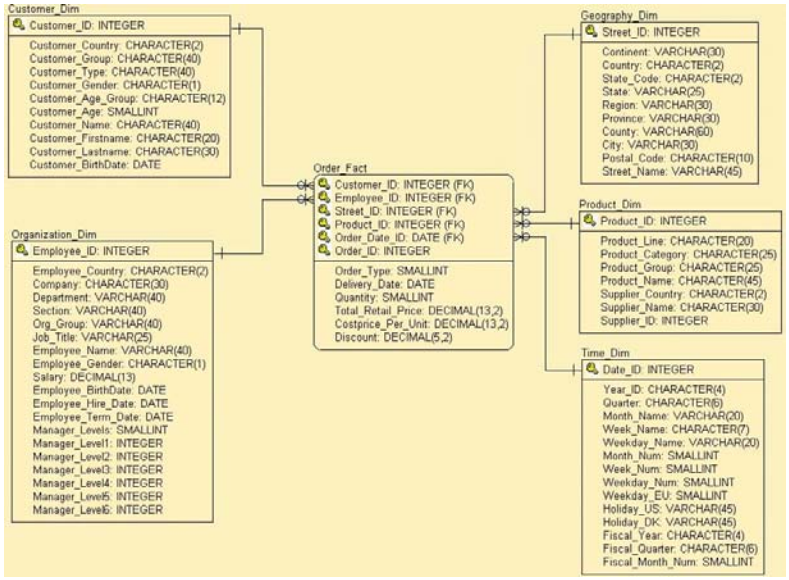

Figure 1. Orion Star Cube Dimensional Data Model

CREATING A CUBE WITH PRIVATE DIMENSIONS

The following code will create a cube with two private dimensions: Time and Customers. Each dimension has a definition block within the PROC OLAP code. In addition, this cube has a global calculated member on the Customers dimension. If you wanted to include either of these dimensions, Time or Customers, in another cube, you would copy this code and change the cube name, fact table, and other cube options. In metadata, each cube would have a reference to separate copies of the dimension metadata and a separate set of physical files.

The same cube can be created by using the Cube Designer in SAS OLAP Cube Studio. After the first cube is built, you could copy the cube definition and then modify the cube by using the Cube Designer. The end result is the same as using PROC OLAP. Each cube has a reference to separate copies of the dimension metadata and a separate set of physical files.

```
OPTIONS VALIDVARNAME=ANY;
OPTIONS FMTSEARCH = (ORDATA);
LIBNAME ORDATA BASE "c:\ordata";
PROC OLAP CUBE = "star"
          PATH = 'c:\ordata\cubes'
          FACT = ordata.order_fact
         EMPTY_CHAR = ' ' EMPTY_NUM = '.';
METASVR OLAP_SCHEMA = "SASApp - OLAP Schema";

DIMENSION Time TYPE=TIME FACTKEY=order_date
      DIMTBL = ordata.time_dim DIMKEY = date
      HIERARCHIES = (Time);

      HIERARCHY Time ALL_MEMBER = 'All Time' CAPTION = 'Year-Week-Day'
         LEVELS = (year week date) DEFAULT;

      LEVEL Year TYPE = YEAR CAPTION = 'Year';
      LEVEL Week TYPE = WEEKS CAPTION = 'Week Name (YYYY-WW)';
      LEVEL Date TYPE = DAYS FORMAT = MMDDYY10. CAPTION = 'Date';

DIMENSION customers  FACTKEY = customer_id
      DIMTBL = ordata.customer_dim DIMKEY = customer_id
      HIERARCHIES = (customers);

      HIERARCHY customers ALL_MEMBER = 'all customers'
         LEVELS = (customer_group customer_type customer_id) DEFAULT;

      LEVEL Customer_Group CAPTION =  'Customer Group';
      LEVEL Customer_Type  CAPTION =  'Customer Type';
      LEVEL Customer_ID    CAPTION =  'Customer Id';

       PROPERTY custage
          LEVEL = customer_id COLUMN = customer_age
          CAPTION = 'Customer Age';

MEASURE quantitysum STAT = sum COLUMN = quantity
       CAPTION = 'Sum of Quantity' FORMAT = comma14.;

AGGREGATION customer_group customer_id customer_type date
            week year / NAME = 'default';

DEFINE Member '[star].[customers].[all customers].[Gold plus Club Members]' AS
   '[customers].[all customers].[Orion Club Gold members] +
    [customers].[all customers].[Orion Club members],
    FORMAT_STRING="BEST15."';
```

CREATING A CUBE WITH A SHARED DIMENSION

Creating the Shared Dimension

Instead of creating the customer dimension within each cube as a private dimension, we now want to create the dimension once and reuse it across cubes. First, we must create the dimension that will be shared across cubes. To create this dimension as a shared dimension, you can use the Shared Dimension Designer in SAS OLAP Cube Studio or submit the PROC OLAP code to create the dimension.

The PROC OLAP code used to create the Customer dimension as a shared dimension is shown below. This code will

create both the metadata and the physical files for the shared dimension.

```
OPTIONS VALIDVARNAME=ANY;
OPTIONS FMTSEARCH=(ordata);
LIBNAME ordata BASE "c:\ordata";
PROC OLAP ;
METASVR OLAP_SCHEMA = "SASApp - OLAP Schema";

DIMENSION "/Shared Data/SASApp - OLAP Schema/customers"
       SHARED
       PATH="c:\ordata\sharedDimension"
      DIMTBL = ordata.customer_dim DIMKEY = customer_id
      HIERARCHIES = (customers)
      EMPTY_CHAR=' ';

      HIERARCHY customers ALL_MEMBER = 'all customers'
         LEVELS = (customer_group customer_type customer_id)DEFAULT;

      LEVEL customer_group CAPTION =  'Customer Group';
      LEVEL customer_type  CAPTION =  'Customer Type';
      LEVEL customer_id CAPTION =  'Customer Id';

      PROPERTY custage
         LEVEL = customer_id COLUMN = customer_age
         CAPTION = 'Customer Age';
```

These are the key differences between the code to create the private dimension and the code to create the shared dimension:

- For a shared dimension, the PROC OLAP statement requires no keywords.
- The keyword, SHARED, must be used in the DIMENSION statement. This indicates that the dimension will be shared.
- The DIMENSION statement must also include a PATH keyword. This specifies where the physical files for the dimension will be stored.
- When creating the shared dimension, the keyword, FACTKEY, is not used. Only the DIMTBL and DIMKEY keywords are required.
- The dimension will be associated with the OLAP schema that is specified in the METASVR statement.
- Shared dimensions are contained in a folder in the SAS Folders tree. You can specify which folder will contain the dimension by including the folder name in the name field of the dimension, as shown above. If no folder is specified, the shared dimension will be contained in the same folder as the OLAP schema.
- Other options, such as EMPTY_CHAR and NONUPDATEABLE, may be specified in the DIMENSION statement. For SAS 9.3, the options for multiple language support are not supported for shared dimensions.
- All member properties associated with the levels in this dimension need to be defined in this code.
- Note that the DEFINE statement for the global calculated member is not included in the code to create the shared dimension. Global calculated members are defined with the cubes that use the shared dimension.

SAS OLAP Cube Studio includes a new wizard, Shared Dimension Designer, to create shared dimensions. In the first panel, you provide the general information about the shared dimension.

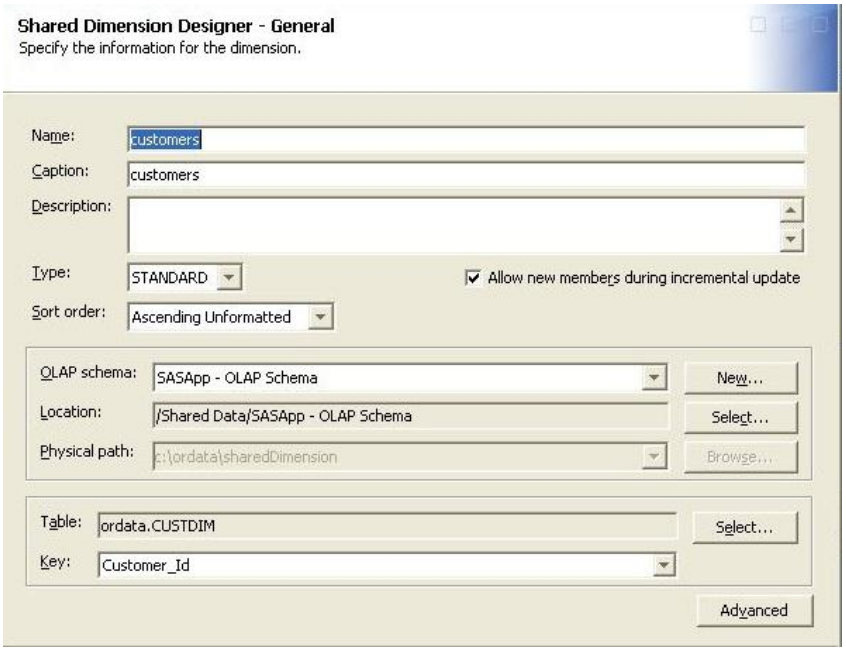

Figure 2. Shared Dimension Designer – General

The second panel of the designer allows you to specify the levels and their attributes.

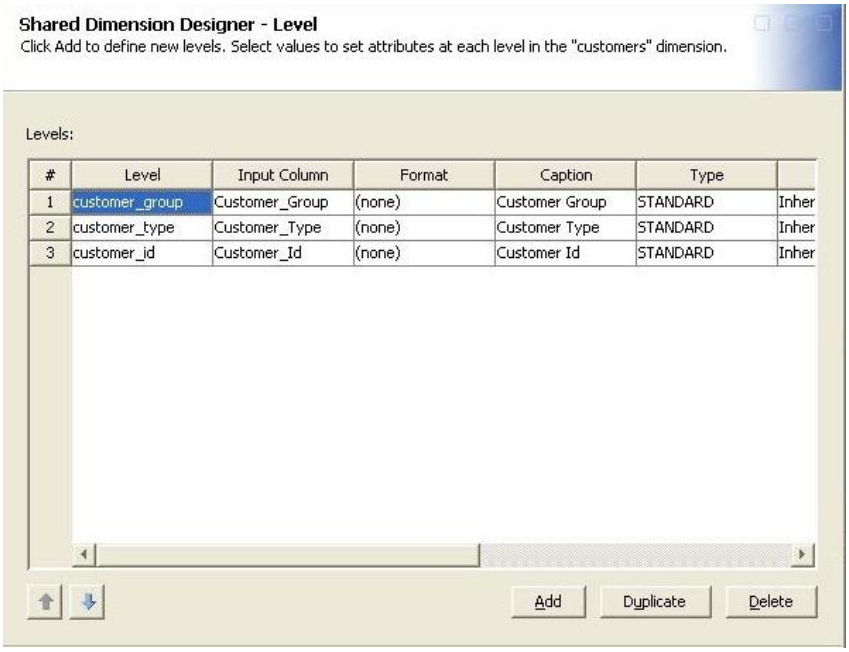

Figure 3. Shared Dimension Designer – Levels

The third panel of the designer allows you to create the hierarchies for the dimension as well as specify member properties and GIS map information.

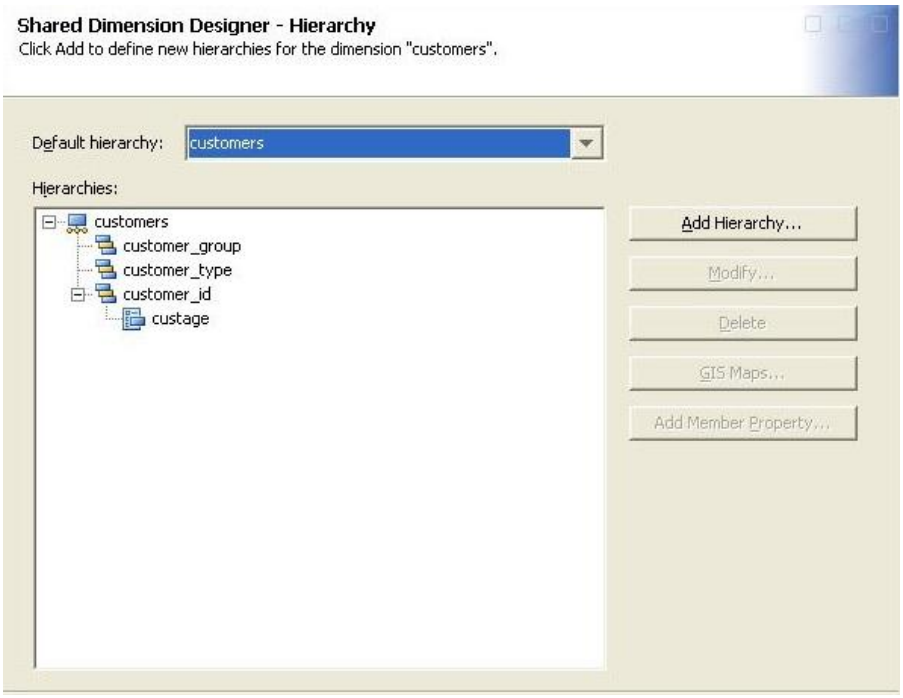

Figure 4. Shared Dimension Designer – Hierarchy

Using the Shared Dimension in a Cube

To use this dimension within a cube, you specify the USE_DIMENSION statement, as shown below, in your cube definition or select the shared dimension within the Cube Designer of SAS OLAP Cube Studio.

```
OPTIONS VALIDVARNAME=ANY;
OPTIONS FMTSEARCH = (ORDATA);
LIBNAME ORDATA BASE "c:\ordata";
PROC OLAP CUBE = "star"
          PATH = 'c:\ordata\cubes'
          FACT = ordata.order_fact
         EMPTY_CHAR = ' ' EMPTY_NUM = '.';
METASVR OLAP_SCHEMA = "SASApp - OLAP Schema";

DIMENSION Time TYPE = TIME FACTKEY = order_date
      DIMTBL = ordata.time_dim DIMKEY = date
      HIERARCHIES = (Time);

      HIERARCHY Time ALL_MEMBER = 'All Time' CAPTION = 'Year-Week-Day'
         LEVELS = (year week date) DEFAULT;

      LEVEL Year TYPE = YEAR CAPTION = 'Year';
      LEVEL Week TYPE = WEEKS CAPTION = 'Week Name (YYYY-WW)';
      LEVEL Date TYPE = DAYS FORMAT = MMDDYY10. CAPTION = 'Date';

USE_DIMENSION customers  FACTKEY = customer_id;

MEASURE quantitysum STAT = sum COLUMN = quantity
       CAPTION = 'Sum of Quantity'
       FORMAT = comma14.;

AGGREGATION customer_group customer_id customer_type date
            week year / NAME = 'default';

DEFINE Member '[star].[customers].[all customers].[Gold plus Club Members]' AS
   '[customers].[all customers].[Orion Club Gold members] +
    [customers].[all customers].[Orion Club members],
    FORMAT_STRING="BEST15."';
```

These are the key differences between the code to create the cube using the private dimension and the code to use the shared dimension in a cube:

- The USE_DIMENSION statement specifies a shared dimension to be included with this cube.
- If the shared dimension is in a different OLAP schema from the cube, the OLAP_SCHEMA keyword must be used in the USE_DIMENSION statement. In this example, the shared dimension is in the same OLAP schema as the cube so the keyword is not used.
- With the USE_DIMENSION statement, only the FACTKEY keyword is required. The DIMKEY and DIMTBL keywords specified in the code to create the shared dimension identify the key column and dimension table for the shared dimension. The FACTKEY keyword identifies the corresponding column in the fact table.
- The levels for the shared dimension can still be used in aggregation statements for the cube.
- Though not shown here, levels from the shared dimension can be used to create measures with the NUNIQUE statistic.
- The DEFINE statement that defines a global calculated member for the Customer dimension is included with this cube. If this calculation needs to be included with other cubes, those cubes will need additional DEFINE statements.

To select the dimension for use in the Cube Designer, open the Cube Designer for the cube and navigate to the Dimensions panel. If you are replacing a private dimension, you will need to select the dimension and delete it. In our example, the private Customer dimension has been deleted and the cube still has the private Time dimension. To include a shared dimension, select the Use Shared Dimensions button.

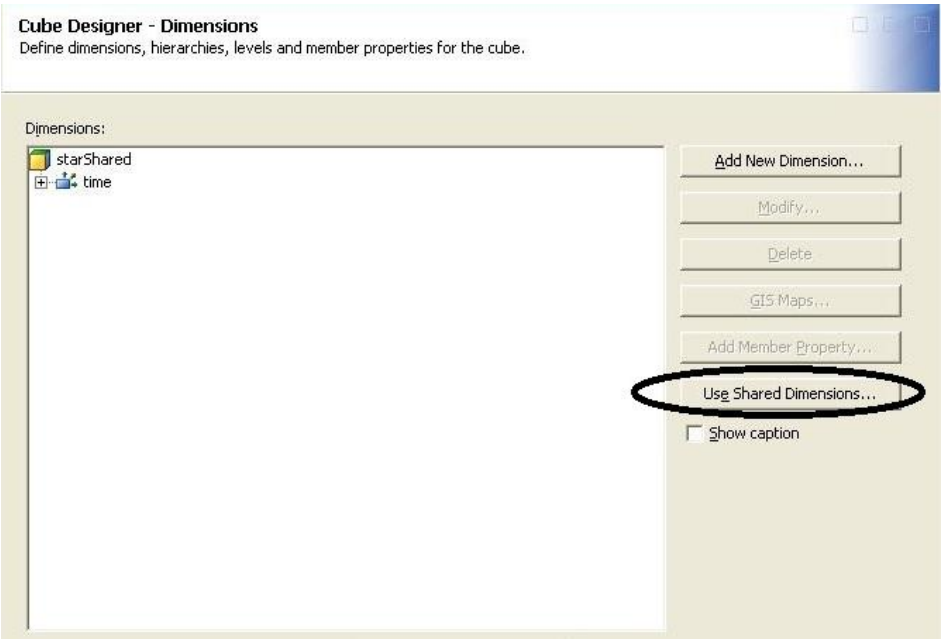

Figure 5. Cube Designer – Dimensions

The Use Shared Dimensions window displays all the shared dimensions defined in metadata along with those currently used by this cube. You can select a shared dimension from the available tree and move it to the selected table to include the shared dimension in the cube. After moving the shared dimension to the selected table, you will need to select the column to be used as a fact key. After selecting a shared dimension to use, the dimension, its levels, and hierarchies will be checked for name compatibility within the cube. All hierarchies, levels, and member properties defined in the shared dimension will be included in the cube.

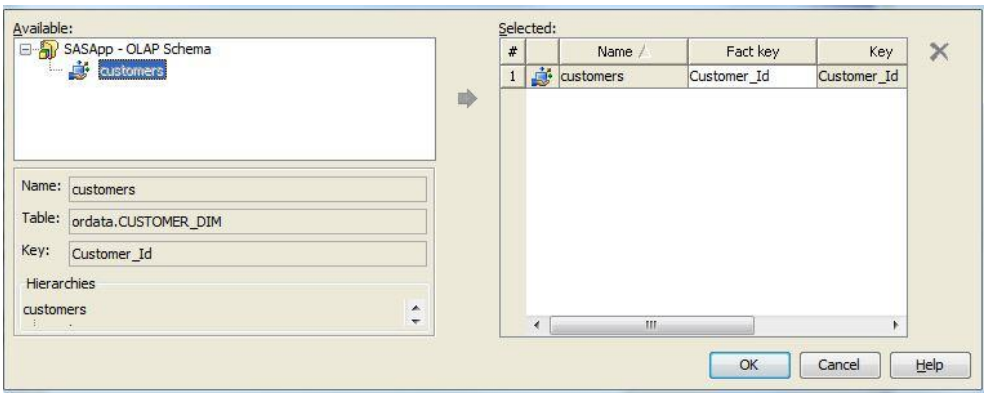

Figure 6. Use Shared Dimensions

Now you can save the metadata and build the cube. In order to build a cube that uses a shared dimension, the shared dimension must be defined in metadata and it must be built. If the shared dimension does not have physical files stored on disk, the cube cannot be built. You will be able to save the cube definition with the reference to the shared dimension, but you will not be able to build the cube.

Shared Dimensions in SAS OLAP Cube Studio Trees

Once the metadata for a shared dimension is created, the shared dimension will be displayed in both the Folders tree and the Inventory tree of SAS OLAP Cube Studio. In the Inventory tree, there is a new category node, Shared Dimensions. Each shared dimension is displayed here and can be expanded to show its levels and hierarchies. If a shared dimension is used by a cube, the shared dimension will be displayed with the other dimensions of the cube but with a different icon. In the Folders tree, the shared dimension will be displayed under the folder that contains it. Like cubes, a shared dimension cannot be expanded in the Folders tree.

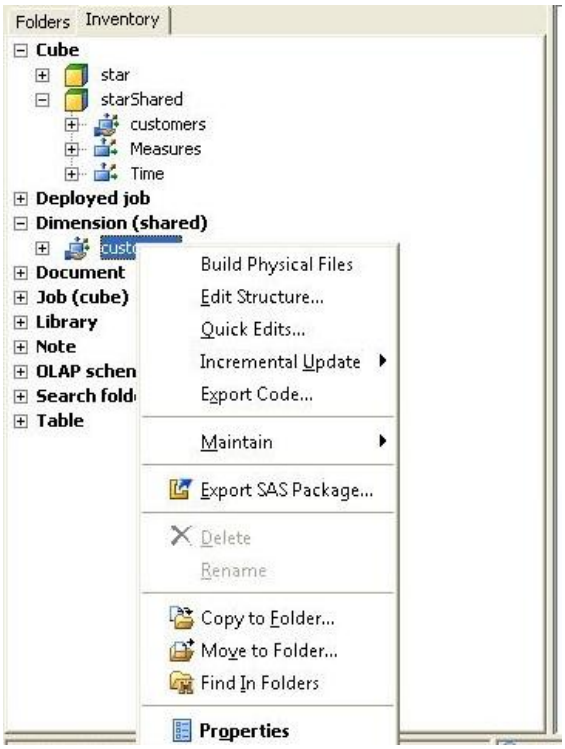

Figure 7. Shared Dimensions in the Cube Studio Inventory Tree

MAINTENANCE TASKS

Applying Permission Conditions to a Cube with a Private Dimension

Permission conditions are applied to private dimensions within the context of a cube. To apply the permission conditions, you navigate to the dimension within the cube and open the property sheet for the dimension. In the Authorization tab, select the user or group whose access you want to limit and then explicitly grant Read. The Add Authorization button will become available. Selecting this button will open the Add Authorization window where you can build the permission condition. The permission condition is applied to the dimension only within this cube.

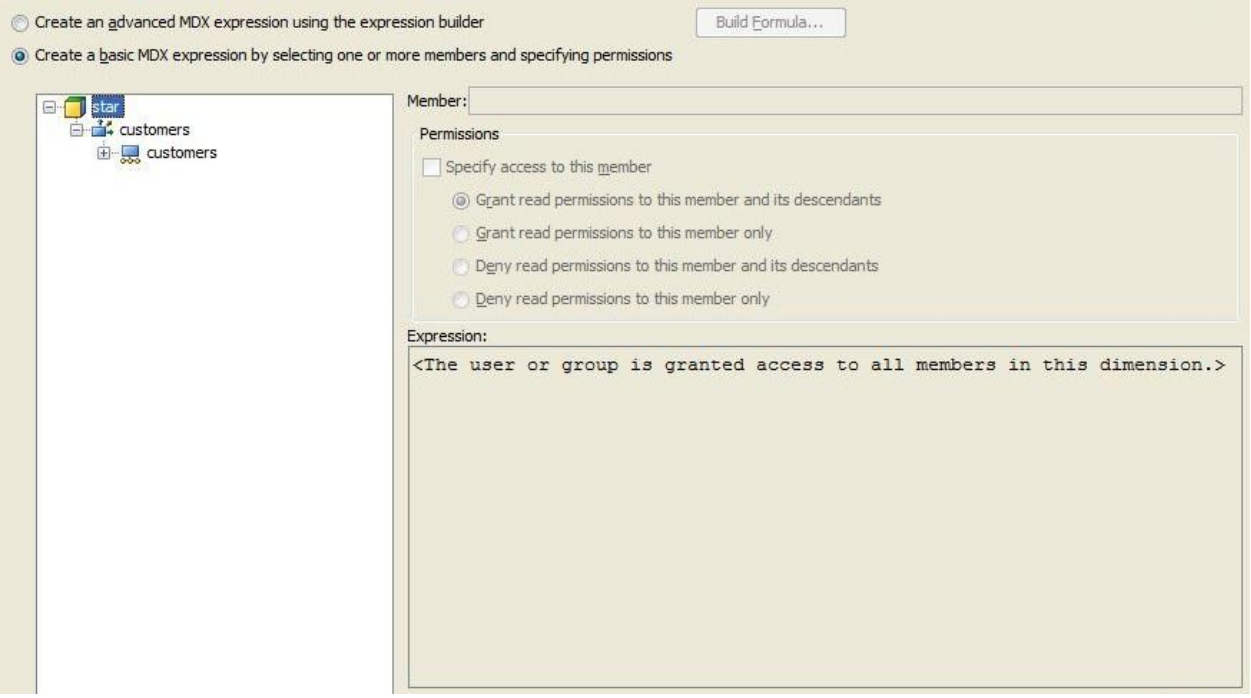

Figure 8. Add Authorization Window for a Private Dimension

Applying Permission Conditions to a Shared Dimension

Like a private dimension, you can apply permission conditions to the members of a shared dimension. Permission conditions can be added at any time to the shared dimension even if no cube currently uses it. To apply the permission conditions, navigate to the shared dimension and open the property sheet. In the Authorization tab, select the user or group whose access you want to limit and then explicitly grant Read. The Add Authorization button will become available. Selecting this button will open the Add Authorization window where you can build the permission condition. They will be used the next time a cube that uses this shared dimension is queried. The same permission conditions will be used for each cube that uses this shared dimension without having to edit each individual cube.

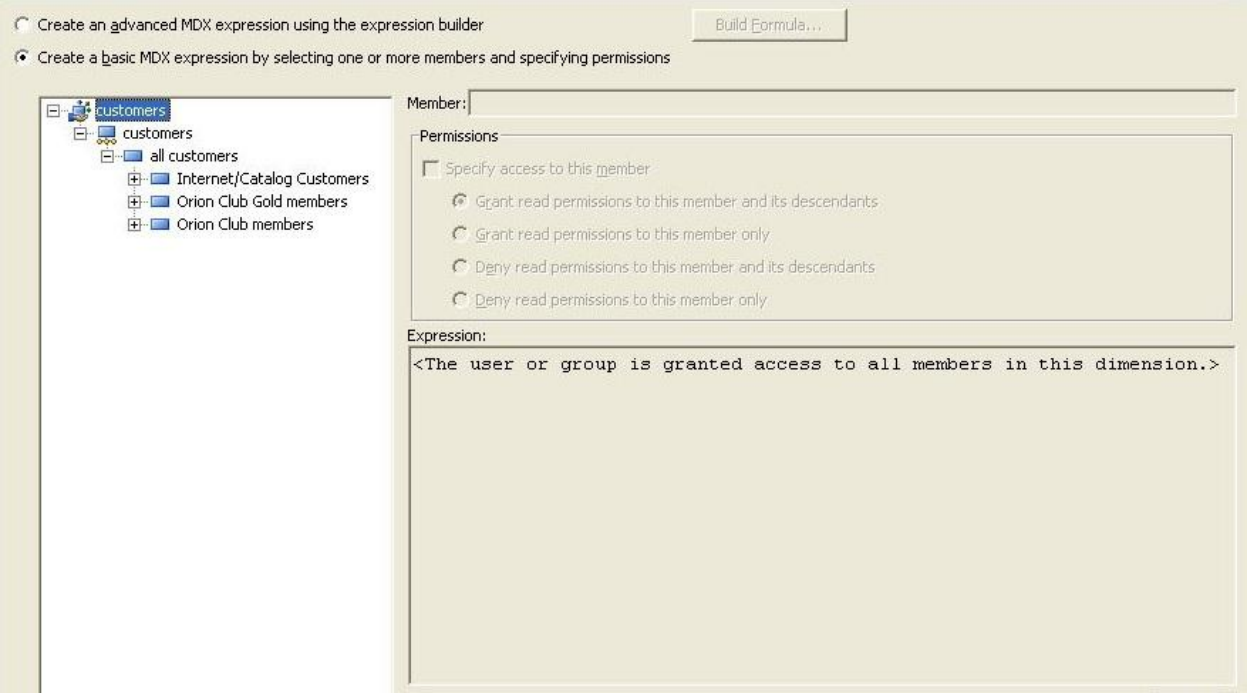

Figure 9. Add Authorizations to a Shared Dimension

Incremental Updating a Cube with a Private Dimension

When you incrementally update a cube, you add new data and possibly new members to the cube. To add new members and data to a private dimension in a cube, you run PROC OLAP code with the ADD_DATA option specifying the table that contains the new facts. If the cube was loaded from a star schema, you may also specify a table that contains the new members for your dimensions. The code below shows the update for a cube with a private dimension. New members will be added to the customers dimension from the CUSTDIMUPD table and matching fact data will be added from the ORDFACTUPD table.

```
/* update a cube with private dimensions */
OPTIONS VALIDVARNAME=ANY;
LIBNAME ordata BASE "c:\ordata";
PROC OLAP CUBE = "/Shared Data/SASApp - OLAP Schema/star"
   DATA = ordata.ORDFACTUPD ADD_DATA UPDATE_INPLACE ;
   METASVR OLAP_SCHEMA = "SASApp - OLAP Schema";
   DIMENSION customers DIMTBL = ordata.CUSTDIMUPD ;
RUN;
```

SAS OLAP Cube Studio also provides a wizard for updating the cube. In the first panel, you specify the table containing the new facts.

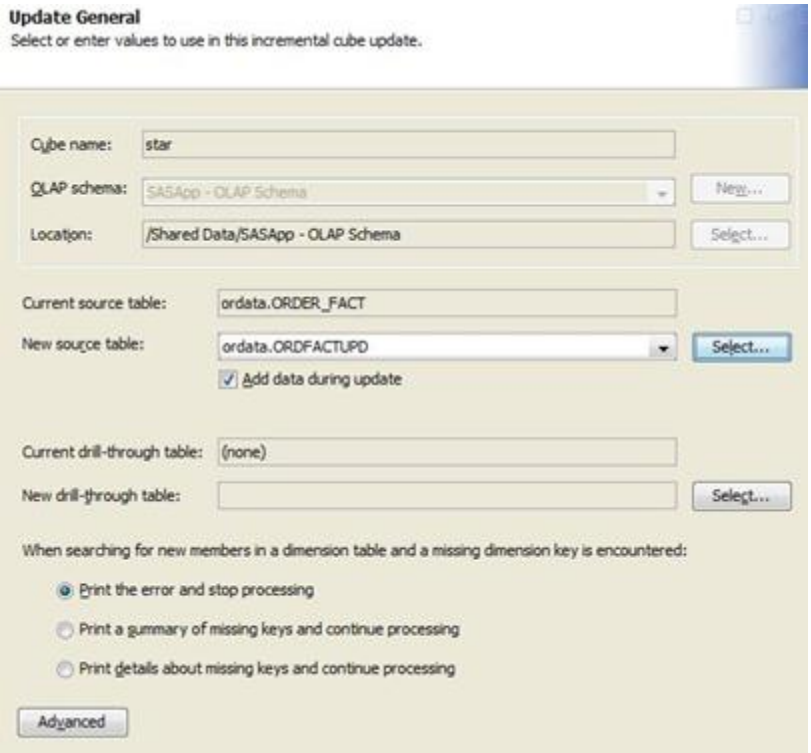

Figure710. Cube Incremental Update – General

In the second panel, you specify the tables containing new member information for the private dimensions.

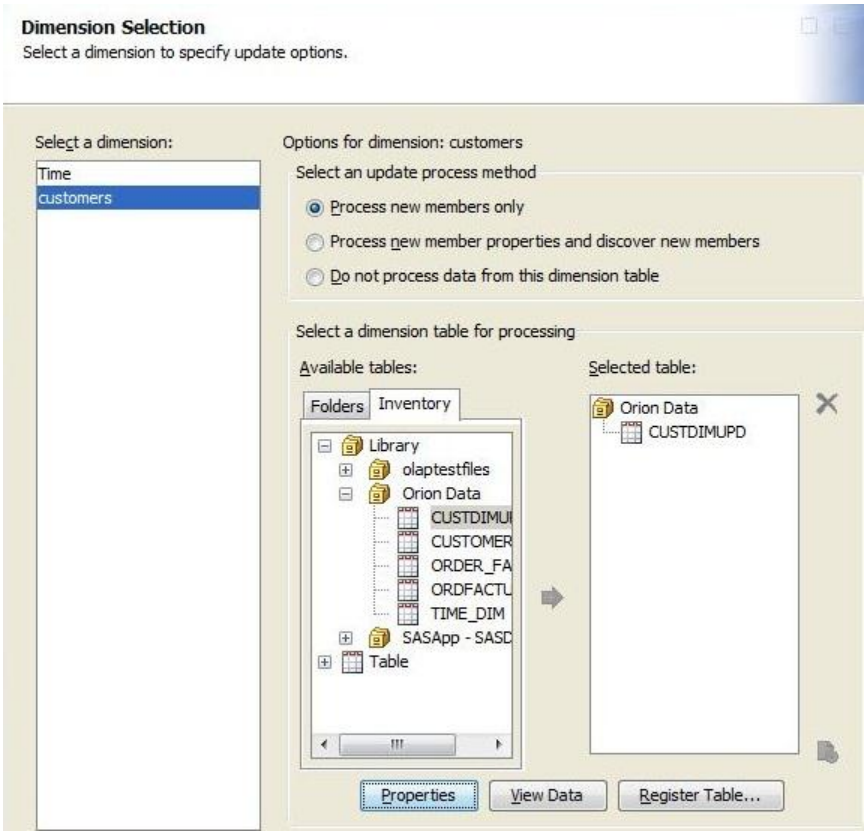

Figure 11. Cube Incremental Update – Dimensions

Incremental Updating a Shared Dimension

Incrementally updating a shared dimension will add new members to the physical files of the shared dimension. In order for those new members to be used by the cubes that use the shared dimension, new fact data must also be added to the cubes after the new members have been added to the shared dimension. This two-step process can be completed by either submitting PROC OLAP code or using the wizards in SAS OLAP Cube Studio. The PROC OLAP code shown below includes both steps.

```
/* Step 1: add new members to the shared dimension */
OPTIONS VALIDVARNAME=ANY;
LIBNAME ordata BASE "c:\ordata";

PROC OLAP;
   METASVR OLAP_SCHEMA = "SASApp - OLAP Schema";
   DIMENSION "/Shared Data/SASApp - OLAP Schema/customers"
      SHARED
      UPDATE_DIMENSION = MEMBERS
      DIMTBL           = ordata.CUSTDIMUPD;
   RUN;

/* Step 2: update the cube with matching fact data */
OPTIONS VALIDVARNAME=ANY;
LIBNAME ordata BASE "c:\ordata";
PROC OLAP CUBE = "/Shared Data/SASApp - OLAP Schema/star"
   DATA = ordata.ORDFACTUPD ADD_DATA UPDATE_INPLACE ;
   METASVR OLAP_SCHEMA = "SASApp - OLAP Schema";
RUN;
```

These are the key differences to note in the code.

- In step 1, the ADD_DATA and UPDATE_INPLACE keywords are not included in the PROC OLAP statement. Updating the dimension with new members does not add new data to any of the cubes, and the update of a shared dimension is always done in-place. The cubes that use the shared dimension may remain online as new members are added to the physical files of the shared dimension.
- In step 1, the SHARED keyword is used in the DIMENSION statement to indicate you are updating a shared dimension.
- In step 1, the DIMENSION statement uses the fully qualified name of the shared dimension, including the folder that contains the shared dimension. This is optional. If not specified, PROC OLAP will use the OLAP schema specified in the METASVR statement to find the dimension definition.
- In step 2, the only difference is that the DIMENSION statement has been removed. Adding new members to the customer dimension was handled in step 1.

After step 1 has finished adding new members, the new members are available for query in the cubes that use the shared dimension. However, until step 2 has finished adding the new facts to the cubes themselves, the new members will show no new data in queries. There is no need to refresh or rebuild the cube. The new members and new fact data will be available the next time the cube is opened for query.

To complete these two steps in SAS OLAP Cube Studio, you first use a new wizard to update the shared dimension. This wizard has a single panel where you specify the table that contains new members for the shared dimension.

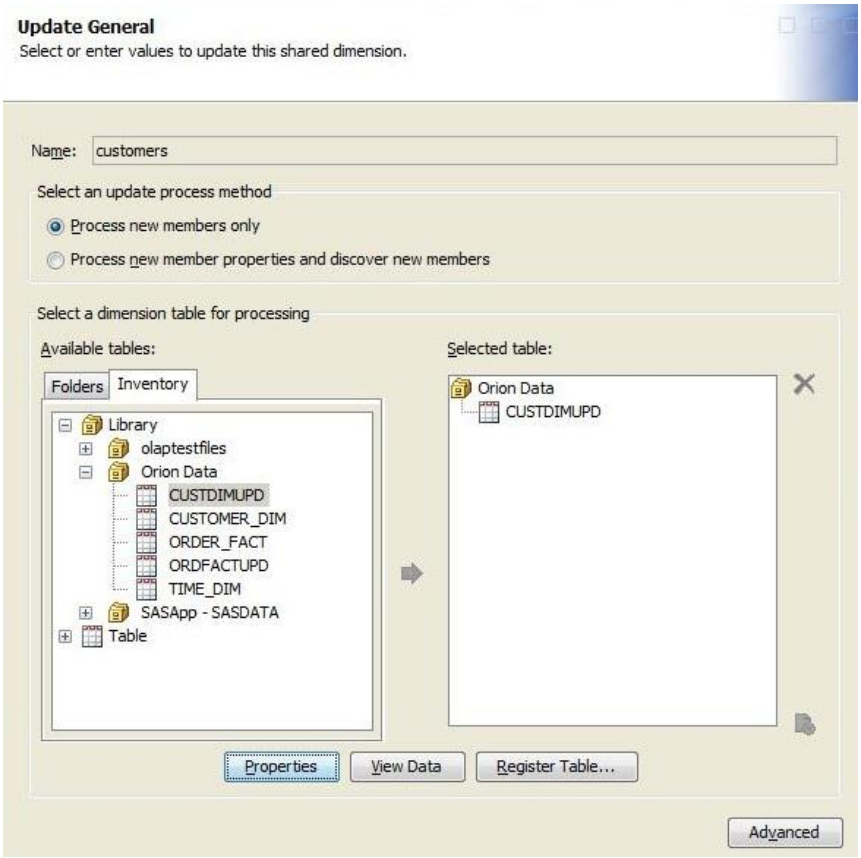

Figure 12. Shared Dimension Incremental Update

After using this wizard to update the shared dimension, you would use the same windows as described in the section, Incremental Update of a Cube with a Private Dimension.

Changing the Structure of a Cube with a Private Dimension

In addition to adding new members and data to a cube, you may need to make structural changes such as adding or removing levels and/or hierarchies. Using PROC OLAP code, you would submit the code to delete the entire cube, make changes to the original code used to create the cube, and then submit that code to build the cube. Using SAS OLAP Cube Studio, you would open the Cube Designer by selecting Edit Structure from the menu. In the Cube Designer, you can make the changes to the structure as needed. When you select Finish in the Cube Designer, the physical files for the cube are deleted leaving the metadata definition in place. After the delete of the physical files is completed, the changes to the metadata are saved. Finally, the cube is rebuilt from the updated metadata definition.

Changing the Structure of a Shared Dimension

When you change the structure of a shared dimension, this change will automatically be reflected across all the cubes that use the shared dimension. In order to accomplish this, all cubes that use the shared dimension must first have their physical files deleted. Once this is done, the shared dimension can be changed and rebuilt. Finally, all the cubes that use the shared dimension can be rebuilt. In SAS OLAP Cube Studio, for each cube that uses the shared dimension, select Maintain -> Delete Physical Files from the menu.

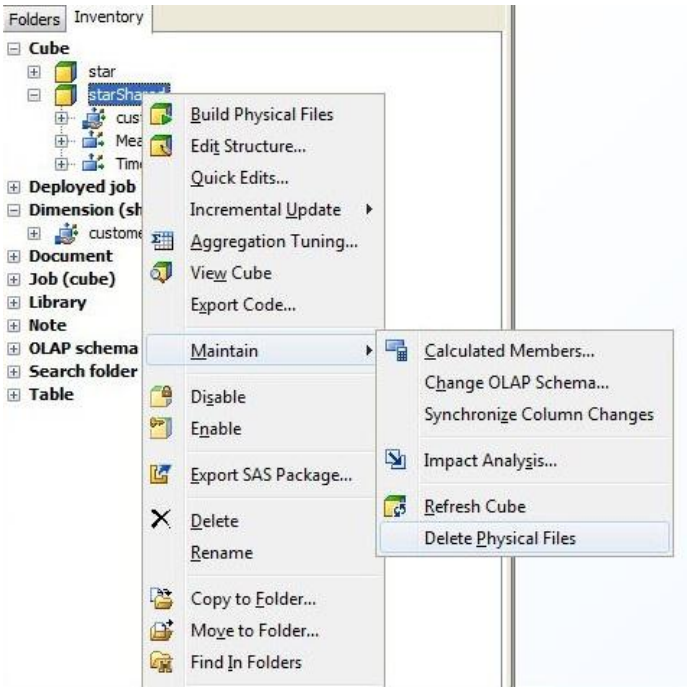

Figure 13. Delete Physical Files Menu Action

Once all cubes have had their physical files deleted, you can select Edit Structure from the menu for a shared dimension. If you select this action prior to deleting all the physical files for all cubes that use the shared dimension, you will receive the following warning:

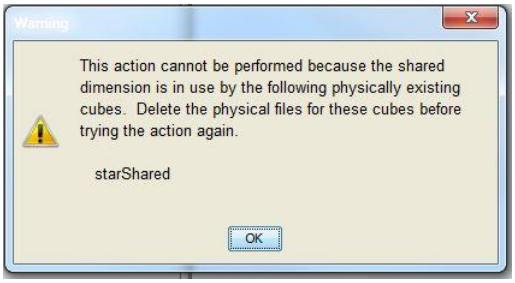

Figure 14. Edit Structure Warning

Selecting the Edit Structure action for a shared dimension will open the Shared Dimension Designer. You can make the changes required in the designer and then select Finish. Before saving changes to the shared dimension structure, the physical files for the shared dimension are deleted leaving the metadata in place. The metadata is then updated and the shared dimension is rebuilt from the updated metadata definition. As part of updating the metadata of the shared dimension, some changes may be made to the metadata definition of the cubes that use the shared dimension. For example, adding or removing a level will change the definition of aggregations in any cube that uses the shared dimension. These changes will be done as the shared dimension metadata is updated.

PROMOTING SHARED DIMENSIONS

In SAS 9.3, shared dimensions can be copied between deployments of SAS software using the promotion tools of the SAS Intelligence Platform. Before promoting your shared dimensions, it is important to understand the dependencies of the shared dependencies. A shared dimension depends on the following:

- OLAP schema it is assigned to
- dimension table and its columns
- library for the dimension table

These dependencies should exist on the target system or they can be included in the same export package as the shared dimension. The Export to SAS Package wizard lets you select the dependencies to include in the same package as the shared dimension.

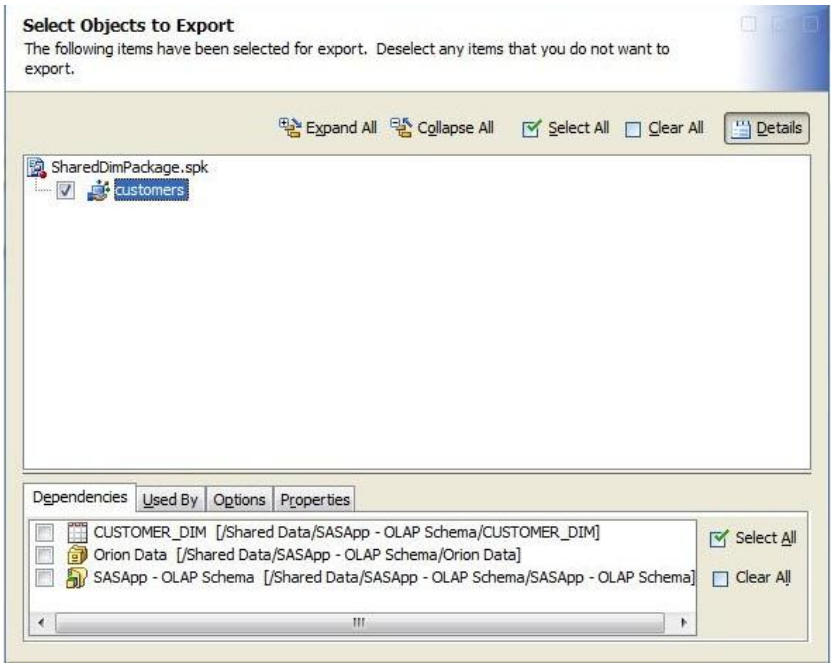

Figure 15. Export to SAS Package – Shared Dimension Dependencies Tab

Optionally, you can include the cubes that use the shared dimension in the same export package as the shared dimension itself. The Used By tab in the Export to SAS Package displays the cubes that use the shared dimension. If you select a cube to include in the export package, you will need to select the cube in the package tree to find its job so that you can also include the job in the export package.

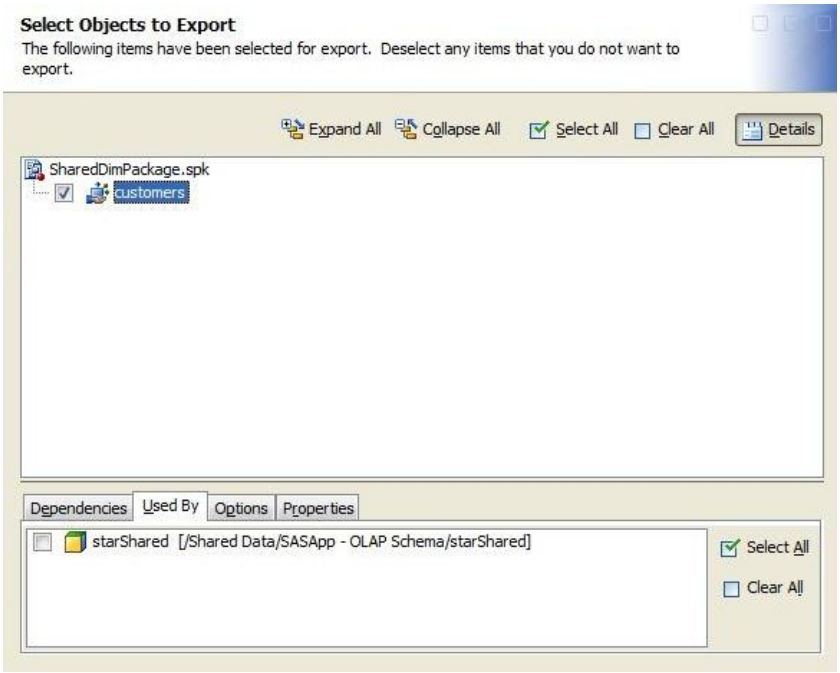

Figure 16. Export to SAS Package – Shared Dimension Used By Tab

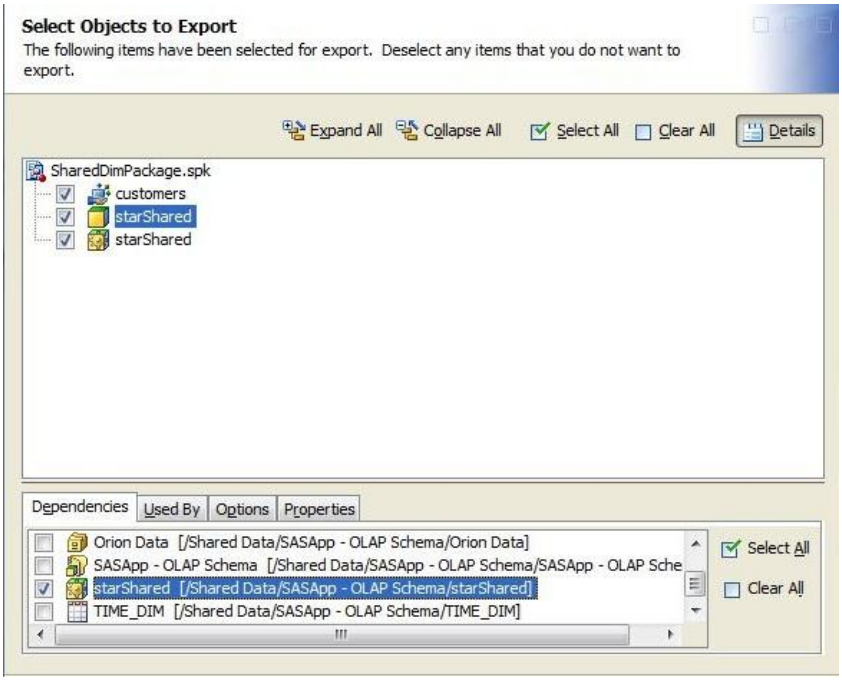

Figure 17. Export to SAS Package – Shared Dimension and Cube

The Import from SAS Package wizard lets you select the objects to import. Depending on the export package selected, you can choose to import the shared dimension and its cubes together or you can choose to import them in separate steps. If you choose to import the cubes later, the associations to the shared dimension will be restored at the point that the cube is imported.

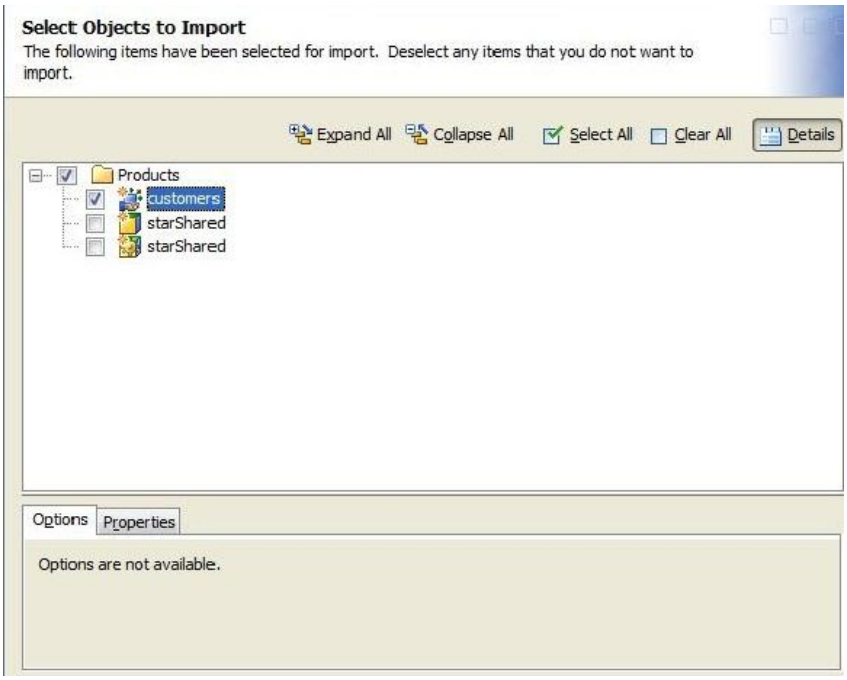

Figure 18. Import from SAS Package – Shared Dimension

The Import from SAS Package wizard will also display prompts that assist you in associating imported objects with the appropriate objects and entities in the target environment. For a shared dimension, this includes the OLAP schema, the dimension table, and the directory path where the physical files for the shared dimension will be stored. Connections between the levels and keys in the shared dimension and columns in the dimension table will be

automatically restored. Appropriate warnings are written to the log for any associations for a shared dimension that cannot be restored in the target environment.

Special Considerations for Promotion of Shared Dimensions

- When importing a shared dimension as a new object, the shared dimension will not have any associations to cubes unless those cubes are included in the same export package and they are imported at the same time as the shared dimension.
- After importing a shared dimension, you must rebuild the shared dimension in the target environment.
- Associations to cubes will be preserved when importing a shared dimension that will overwrite a shared dimension on the target system.
- When overwriting a shared dimension on a target system, modifications to associated cubes may be made to account for changes in the shared dimension. For example, a new level imported with the shared dimension will be added to the default or NWAY aggregation of each cube associated with the shared dimension.
- Since a shared dimension must be rebuilt after it is imported, any cubes that are associated with the shared dimension on the target system will need to be rebuilt.
- If you import a cube that connects to a shared dimension in the target environment, you must build the cube in the target environment.

BEST PRACTICES

- Use a script to incrementally update a shared dimension and its associated cubes. The script could do the following:
  - disable all the cubes, if desired
  - incrementally update the shared dimension to add new members
  - incrementally update the cubes with new facts for the new members
  - enable the cubes, if they had been disabled
- Grant both the Read permission and the ReadMetadata permission for any user who will query cubes that use the shared dimension. Unlike private dimensions, shared dimensions will not inherit authorizations from the cube that the shared dimension is used in.
- Use the SAS Folders tree in SAS® Management Console, SAS OLAP Cube Studio, or SAS® Data Integration Studio to change authorizations on a shared dimension. Remember that authorizations set on a shared dimension will affect all the cubes that use the shared dimension.
- When copying shared dimensions and cubes between deployments of SAS software, try to copy the cubes at the same time as the shared dimensions they use. However, if you need to copy a cube to a target environment that already includes any shared dimensions the cube uses, then it is better to just copy the cube.

CONCLUSION

This paper has introduced the capability of creating and using shared dimensions in SAS OLAP cubes. Shared dimensions ensure consistency across cubes that include them and help reduce maintenance requirements when the dimension needs updating. By stepping through a case study of updating an existing cube to incorporate a shared dimension, we have seen how easy this is to achieve, and discussed how to perform other standard tasks like adding permission conditions and incrementally updating the shared dimension. We have also seen how to promote shared dimensions across SAS environments so they can participate in your standard lifecycle operations.

Shared dimensions offer benefits that SAS OLAP cube administrators will appreciate. With SAS 9.3, they are fully integrated into the SAS OLAP Server and SAS OLAP Cube Studio, and are ready to use. Administrators and cube designers will rest easier knowing that they are helping the environment as well as themselves. In addition to ensuring consistency across all cubes, re-using storage for member data and metadata economizes storage use.

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

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

CONTACT INFORMATION

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

Jana Van Wyk
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919) 531-6914
(919) 677-4444
jana.vanwyk@sas.com
www.sas.com

Ann Weinberger
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
(919)-531-6607
(919) 677-4444
ann.weinberger@sas.com
www.sas.com