Paper 123

Administering OLAP with SAS/Warehouse Administrator(TM)

By Michael Burns, SAS Institute Inc., Austin, TX.

Abstract:

When building an OLAP application, there are a wide variety of strategies that can be used ranging from a simple summary table, or an MDDB with many crossings, to a HOLAP proxy with several associated MDDBs and summary tables. The process of building such a solution and getting it registered in a SAS/EIS(R) metabase can be quite complex. Changing your strategy can be a lot of work.

Now, there is an OLAP Wizard [1] available as an add-in tool for SAS/Warehouse Administrator 1.3 [2] that greatly simplifies the process as well as allows you to change strategies. It builds and maintains an OLAP data group for you, and the associated code generator writes all of the PROC MDDB or PROC SUMMARY code to build the datastores. Finally, an associated metadata exporter registers it all for you in an SAS/EIS metabase.

Background Understanding

This paper assumes that you have some understanding of the SAS/Warehouse Administrator product and OLAP concepts in general, as well as the spiral diagram methodology. For more information on that methodology, refer to SAS/EIS Technical Report: HOLAP Extension, Release 6.12 [3] which is SAS publication 56564. Chapter 4 (**Implementation Strategies**) explains the spiral methodology. SAS HOLAP extensions are available now in conjunction with SAS Consulting Services. For more information, contact your local SAS software representative.

For an understanding of the OLAP Wizard consult its user documentation at OLAP Wizard User Documentation [4].

Input Data

The sample input data used in this demonstration is the Toy Sales Database built for a Data Warehouse training session. The data captures sales of toys across several product lines and manufacturers. It has information about the age and gender of the customers. There are also dimensions involving distribution mechanisms, geography, and promotional campaigns.
The design for the warehouse centered around the star schema shown in Figure 1.

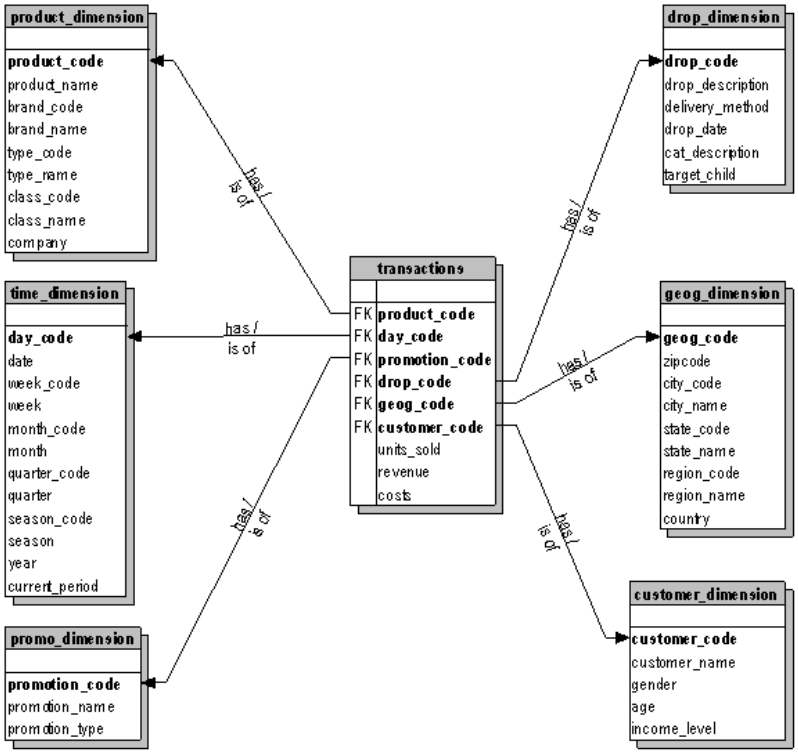
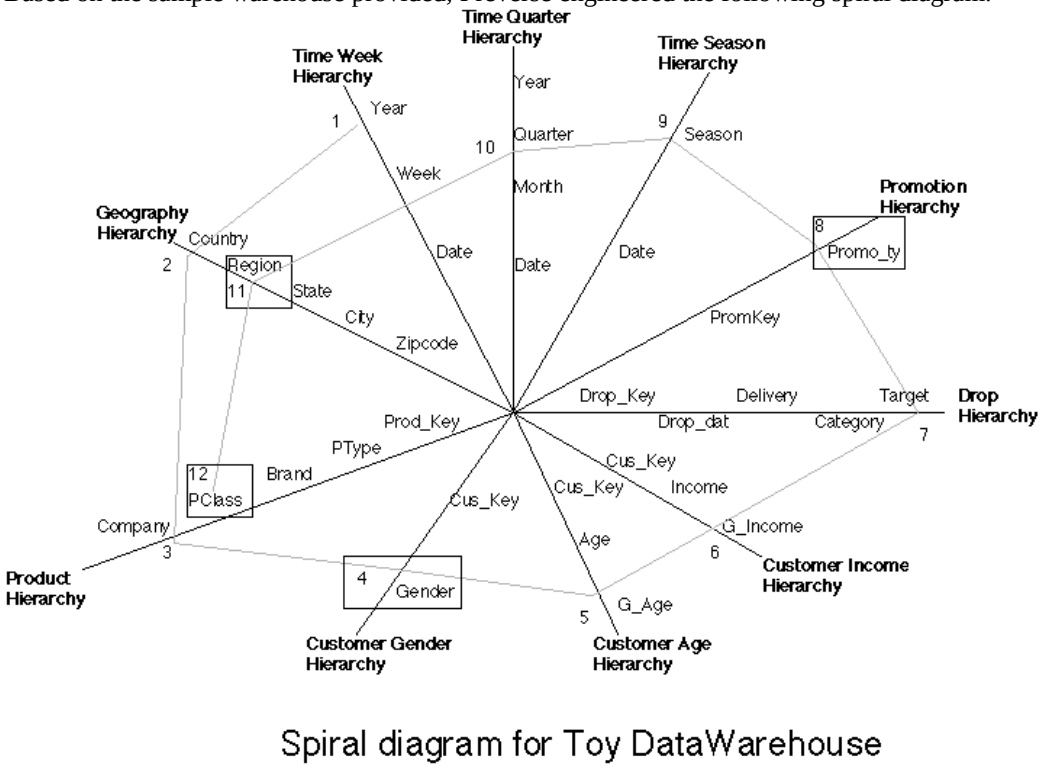


Figure 1: Toy Sales Star Schema

Based on the sample warehouse provided, I reverse engineered the following spiral diagram:



Spiral diagram for Toy DataWarehouse

Figure 2: Toy Sales Spiral Diagram

Dimensions

There are 8 separate dimensions to this data, shown on the diagram moving clockwise from the top. All but the time dimension have only a single drilldown hierarchy. In those cases, the hierarchy will be named the same as the dimension.

- **Time Dimension**
  The designer wanted three different ways to explore the time dimension, so he designed three separate hierarchies:
  - **Time Week Hierarchy** : Year, Week, Date.
  - **Time Quarter Hierarchy** : Year, Quarter, Month, Date.
  - **Time Season Hierarchy** : Season, Date.
- **Promotion Dimension** : Promo_ty, PromKey.
- **Drop Dimension** : Target, Category, Delivery, Drop_dat, Drop_Key.
- **Customer Income Dimension** : G_Income, Income, Cus_Key.
- **Customer Age Dimension** : G_Age, Age, Cus_Key.
- **Customer Gender Dimension** : Gender, Cus_Key.
- **Product Dimension** : Company, PClass, Brand, PType, Prod_Key.
- **Geography Dimension** : Country, Region, State, City, Zipcode.

Crossings

The specifications for the data warehouse called for 12 separate levels of pre-summarization or **crossings**. These crossings are intended to provide quick response for the many desired reports. Each crossing results in a summarization of all of the data for each unique combination of the class variables in that crossing.

```
Year Country Company Gender G_Age G_Income Target Promo_ty Season Quarter Region Pclass
Year Country Company Gender G_Age G_Income Target Promo_ty Season Quarter Region
Year Country Company Gender G_Age G_Income Target Promo_ty Season Quarter
Year Country Company Gender G_Age G_Income Target Promo_ty Season
Year Country Company Gender G_Age G_Income Target Promo_ty
Year Country Company Gender G_Age G_Income Target
Year Country Company Gender G_Age G_Income
Year Country Company Gender G_Age
Year Country Company Gender
Year Country Company
Year Country
Year
```

When looking at the spiral diagram (figure 2), you can visualize each of these crossings as a path along the spiral starting with the first variable named (**Year** in each case) and going inwards ending with the last variable named. Thus, the third crossing is depicted as the path of the spiral from **Year** counter-clockwise around to **Quarter**.

Storage Units

Before the invention of Hybrid OLAP (HOLAP), you had to store all 13 of the crossings in a single MDDB file. Such a file could become quite large and might not fit on your local disk. We could easily try that type of a solution using the OLAP Wizard. I did once and the resultant MDDB file was 59 megabytes.

If we later decided that we wanted to split the crossings across physical stores, it would be easy to do that. One great advantage of the HOLAP data provider is that it does not require storing the NWAY crossing. This is possible because the wizard registers enough information in the metabase for the data provider to access the detail table directly. If the detail table is a star schema implemented as an SQL view, then the provider dynamically builds a view that includes only the fact table and the dimensions that are needed for any particular query. In this example, eliminating the NWAY crossing saves 44 megabytes.

To simplify this demonstration, we will skip the less-efficient MOLAP plan and go straight to the more complex but efficient HOLAP plan.

The HOLAP design calls for four separate physical storage units to house these various crossings:

- **SUM_12 Summary Table** To house the largest crossing with 12 class variables. Since this is a summary table and contains only a single crossing it is depicted on the spiral diagram (figure 2) as a rectangle around the **PClass** column of the **Product Hierarchy**. Just like the crossing that it contains, you should visualize this as starting with **PClass** and spiraling outwards to **Year**.
- **SUM_11 Summary Table** To house the second largest crossing with 11 class variables. This is depicted as a rectangle around **Region**.
- **SUM_10 MDDB** To house a copy of the other 10 crossings. Since this contains many crossings, it was not graphically depicted on the diagram like the summary tables were.
- **SUM_8 Summary Table** To house the crossing with 8 class columns. This is depicted as a rectangle around **Promo_ty**.
- **SUM_4 Summary Table** To house the crossing with 4 class columns. This is depicted as a rectangle around **Gender**.

Obviously, we need a proxy MDDB to pull it all together into a single logical MDDB.

Data Setup

1. Make sure you are setup to run SAS/Warehouse Administrator, the HOLAP extensions to SAS/EIS, and the OLAP Wizard as described in their respective documentation referred to above. This assumes you have a Windows95 or Windows/NT machine with The SAS System version 6.12 installed (including SAS/EIS and SAS/Warehouse Administrator 1.3) in the directory "C:\SAS".
2. Download the self extracting zip OLAPWizardToyDemo.exe [5] and run it. Accept the default extraction location of C:\OLAPWizardToyDemo.
3. You may need to make minor changes to the included autoexec.sas, config.sas and "SAS 6.12" shortcut depending on your local setup.
4. The provided autoexec.sas sets up the appropriate libnames and initiates the DW command. In the SAS/Warehouse Administrator folder you should find a warehouse environment named "**OLAP Wizard Toy Demo Warehouse Env**". In that environment you should find a warehouse named "**Toy Sales Warehouse**" and an Operational data group named "**Operational Data**". Expand the warehouse and you should see a subject named **Toy Sales**.
5. In the **Toy Sales** subject you should find a Detail Logical Table (DLT) that already contains details tables that correspond to the operational data. The process editor is already setup to load those detail tables from the operational data while applying appropriate formats and labels. Finally, the load step for the detail logical table itself is set to build an SQL view of the detail tables. That view implements the star schema.
6. You should select each of the detail tables and **Run** them. Finally **Run** the DLT itself. Verify that the steps all ran properly and that you can view the **DATA.STAR** view.

Using the Wizard

1. Select the **Toy Sales** subject. Then use **Tools-Add-ins...-OLAP Wizard** pull down menu item. If the OLAP Wizard is the only add-in tool you have registered, then **Tools-Add-ins...** will invoke it without you having to make an additional choice.
2. In the **Introduction** page of the wizard, select **Hybrid OLAP Extensions** and press **Next** .
3. In the **Define class variables** page, select the few remaining Source Columns that need to be designated as Class Variables by using MB1 to click on them:
   - INCOME
   - YEAR
   - ZIPCODE

   Now, move those to the Class Variables list by pressing the single black arrow pointing to the right.
4. **Assign cardinality** values to the class variables that are on the spiral starting with the outermost as 1 (**Year**) and spiraling inward.
   I did not have specific cardinalities for this data and assumed that the designer had already made the correct design decisions and chose an appropriate spiral. As a result, I simply numbered the class variables on the spiral from the outside, starting with 1 and ending with 12. Knowing that the spiral crossing algorithm in the wizard used ascending cardinalities, I chose to use these simple numbers for cardinalities.
5. If you want, you can also modify the **Sort Order** for any of the variables. When you are done press **Next** .

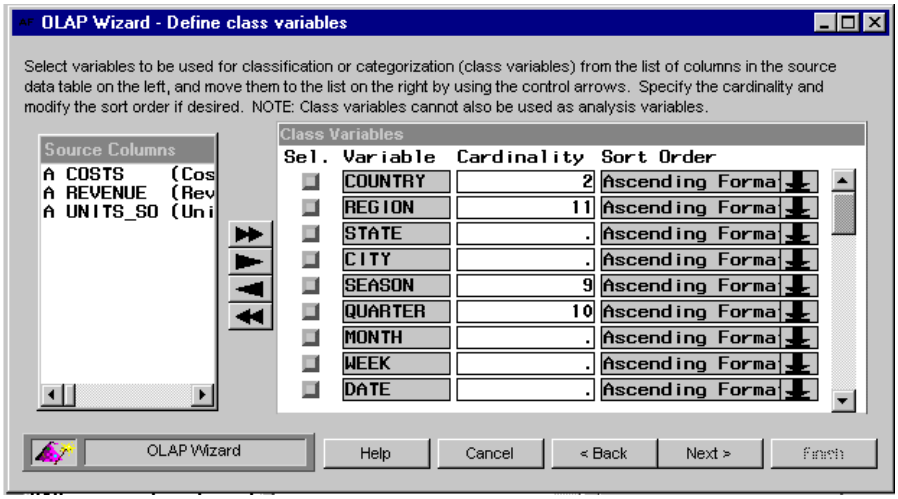


Figure 3: Completed class variables definition screen.

6. In the **Define analysis variables** page of the wizard, the three remaining columns of the input data are automatically selected as Analysis Variables. The only thing you might want to do here is to modify the Base Statistics to be used in the summarization process. However, **SUM** is the default and most sensible value. When you are finished, press **Next** .

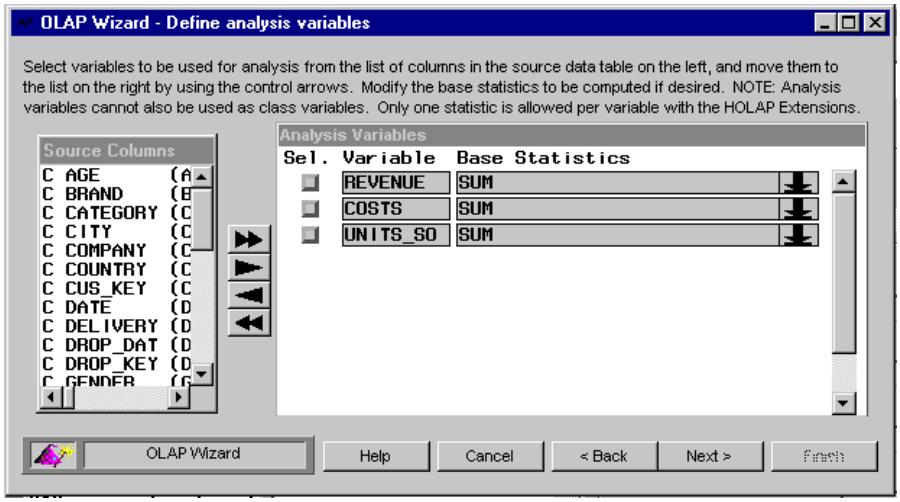


Figure 4: Completed analysis variables definition screen.

7. In the **Define display hierarchies** page of the wizard, you will create a separate hierarchy for each of the rays of the spiral diagram. By default, there is a hierarchy created for you named **H0000001**. You can modify that name by selecting it with MB1 and pressing the **Delete** key 8 times then typing the new name. Start first with **TIME WEEK** by entering its name in place of **H0000001**.When you press **Enter** after entering the new name, the whole row of the **Hierarchies** list should once again be selected as evidenced by the black background around the selection button.

   Now, to select the variables that constitute this hierarchy, simply click on them in the correct order (outside to inside) from the ray on the spiral diagram. In the case of **TIME WEEK**, you would select **YEAR, WEEK, and DATE** in that order. If you mistakenly select another variable, simply click it again to unselect it and remove it from the list of variables being built. After you have selected each of the variables and check the list built, you can move on to the next hierarchy by pressing the **Add** button. In a similar fashion you should create each of the hierarchies:

   - **Time Week** : Year, Week, Date.
   - **Time Quarter** : Year, Quarter, Month, Date.
   - **Time Season** : Season, Date.
   - **Promotion** : Promo_ty, PromKey.
   - **Drop** : Target, Category, Delivery, Drop_dat, Drop_Key.
   - **Customer Income** : G_Income, Income, Cus_Key.
   - **Customer Age** : G_Age, Age, Cus_Key.
   - **Customer Gender** : Gender, Cus_Key.
   - **Product** : Company, PClass, Brand, PType, Prod_Key.
   - **Geography** : Country, Region, State, City, Zipcode.

   When you have completed all 10 hierarchies, press **Next**

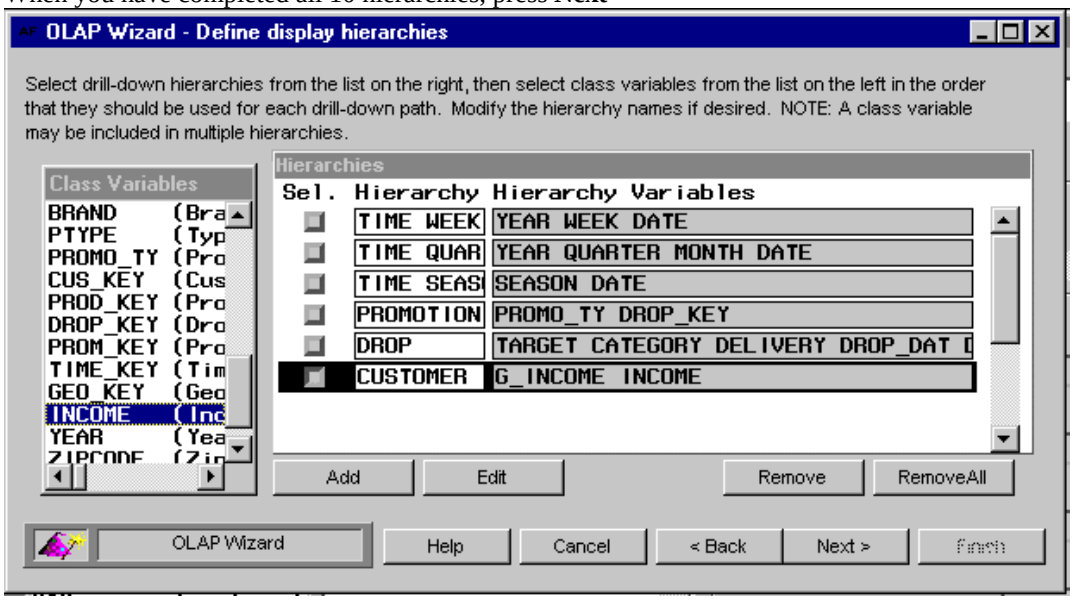


Figure 5: Definition of the Customer Income hierarchy.

8. In the **Define stored crossings** page of the wizard, you can manually or automatically generate an optimal set of crossings. Initially, the wizard will define a single NWAY crossing. Additionally, we need to add the 12 crossings defined by the spiral. To achieve this, simply select **Special** and **Add spiral crossings**. Doing so will result in 13 crossings named **X0000001** to **X0000013,** as such:

```
X0000001 (all class variables).
X0000002 Year Country Company Gender G_Age G_Income Target Promo_ty Season Quarter Region Pclass
X0000003 Year Country Company Gender G_Age G_Income Target Promo_ty Season Quarter Region
X0000004 Year Country Company Gender G_Age G_Income Target Promo_ty Season Quarter
X0000005 Year Country Company Gender G_Age G_Income Target Promo_ty Season
X0000006 Year Country Company Gender G_Age G_Income Target Promo_ty
X0000007 Year Country Company Gender G_Age G_Income Target
X0000008 Year Country Company Gender G_Age G_Income
X0000009 Year Country Company Gender G_Age
X0000010 Year Country Company Gender
X0000011 Year Country Company
X0000012 Year Country
X0000013 Year
```

When you have verified that the crossings are correct, press **Next** .

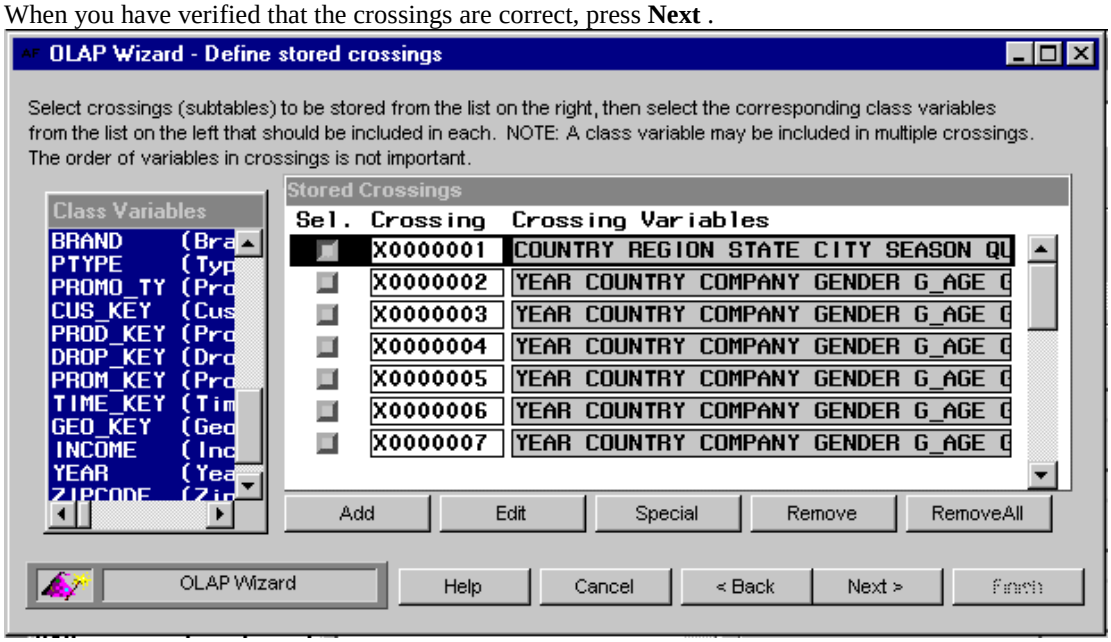


Figure 6: Completed stored crossings definition screen.

9. On the **Define storage Locations** page, you will define the physical storage units and select which crossings are stored in each. Initially a single MDDB named **T0000001** is defined by default.

   Remove it by pressing **Remove.**

   Now, define the required 5 physical stores as described above:

   - **SUM_12 Summary Table** containing **X0000002**.
     To define, press **Add**, then select **Summary Data Set**. That will create a table named **T00000001**, which you should rename to **SUM_12**. Then select **X0000002** from the Crossings list. By default, all analysis variables are selected and you should probably keep it that way. When the Crossings and Analysis Variables are selected properly, press **Add** to move on to the next storage unit.
   - **SUM_11 Summary Table** containing **X0000003**.
   - **SUM_10 MDDB** containing **X0000004** through **X0000013**
     When selecting the crossings, be careful to wait a short time after each selection for the hour-glass sprite to disappear. It takes a little time to process each selection and further selections made while the hour-glass is displayed may not be processed properly because of mouse-event timing problems.
   - **SUM_8 Summary Table** containing **X0000006**.
   - **SUM_4 Summary Table** containing **X0000010**.

   When all storage units are defined, press **Next .**

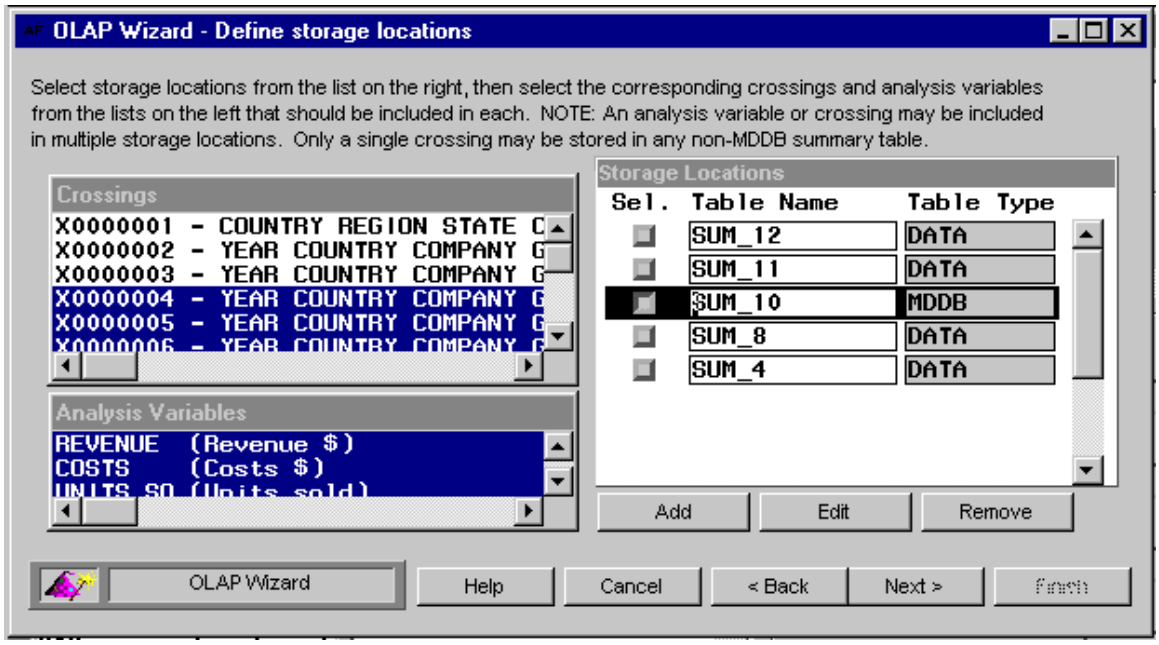


Figure 7: Completed storage locations definition screen.

10. The **Wrap Up** page is the place where you commit all of the metadata needed to represent your new OLAP Group to the SAS/Warehouse Administrator metadata repository. This happens when you press **Finish**. This will take some time because there is a lot of metadata to register and many cross dependencies to establish and validate. Be patient.

Once the metadata is saved, you should see your new OLAP Group appear in the Data Warehouse Environment Explorer view. You can expand it and see an item for the PROXY MDDB and another one for each of the 5 storage units we specified.

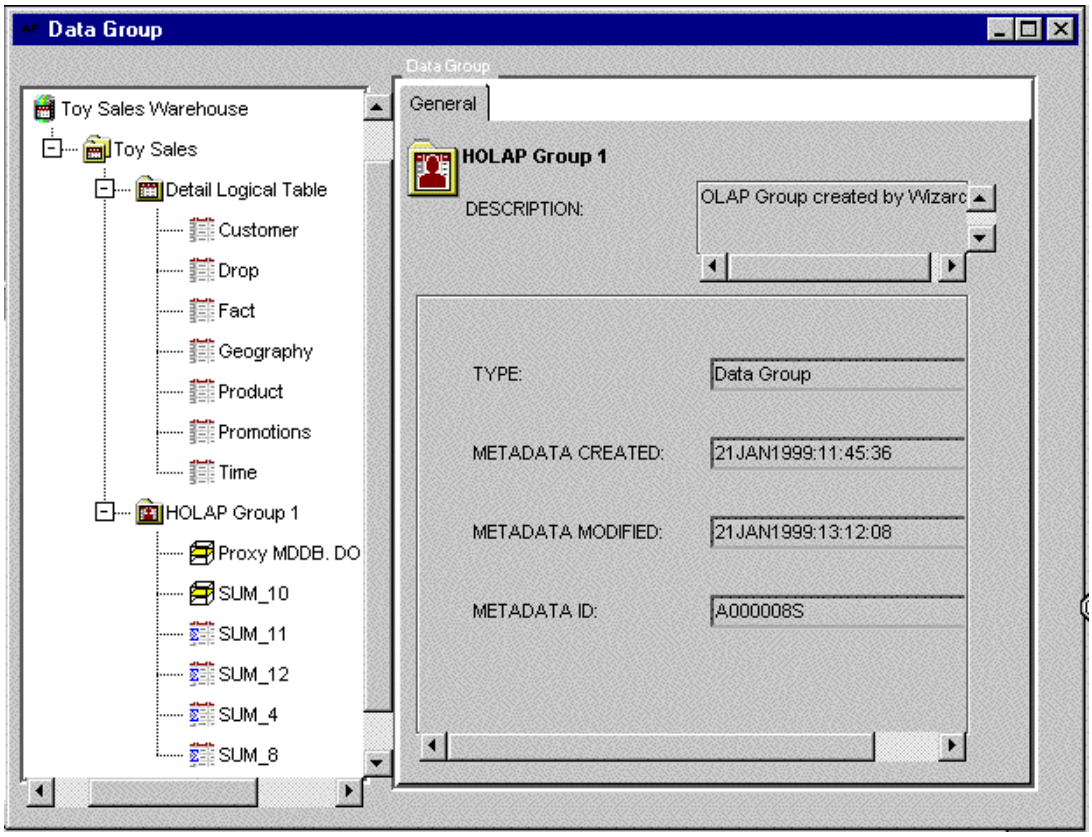


Figure 8: Explorer view with newly defined OLAP Group.

Take a moment now to explore the OLAP Group. In doing so, you will see the considerable amount of work the Wizard did for you. By selecting any member of the group and using **MB3-Properties...**, you can see the properties of the item. Most of the work done by the wizard is defining the many columns needed. You can see this in the **Columns** tab. In that tab, select one of the columns and use **Edit-Column Extensions...**. For each of the analysis columns **COSTS, REVENUE, UNITS_SO** you will see that the **STATISTIC** extended attribute has been defined. For each of the class columns (**Year** for example) you will see **SORTORDER, CARDINALITY, and CLASS** extended attributes. You will also see a separate **CROSSING ...** extended attribute for each crossing the column participates in. If you selected the **PROXY MDDB** to view properties of, you will also see **HIERARCHY ...** extended attributes for each hierarchy a column participates in.

Between the 6 items in the OLAP Group there are over 94 separate column registrations that have been made and about 966 extended attributes.

Build the MDDB and Summary Tables

Following the instructions in the documentation for the OLAP Wizard, you should now define physical storage properties for each of the 6 items in the OLAP Group and create a mapping to the load source and actually load the table.

I would suggest that you create a separate library to house the items of the OLAP Group. Call it **HOLAPGP1**.

You should load them in the following order so that you can actually load one summary table from another rather than loading each one from the entire detail logical table.

1. **Proxy MDDB** physical storage: **HOLAPGP1.PROXY** load from DLT **DATA.STAR**

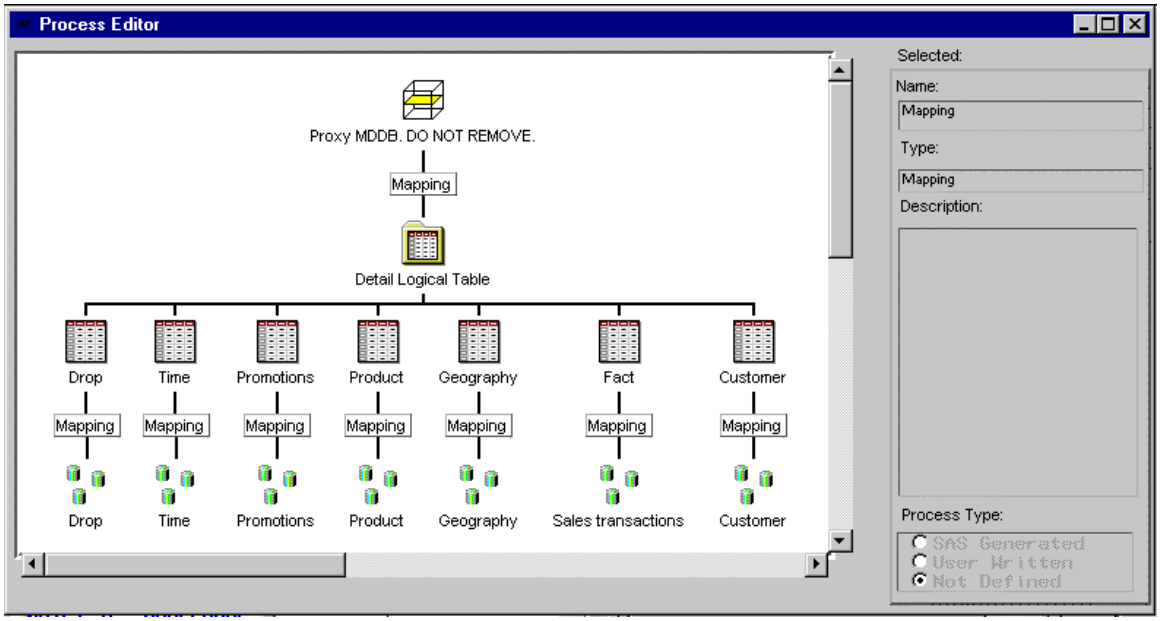


Figure 9: Process editor diagram for PROXY.

2. **SUM_12** physical storage: **HOLAPGP1.SUM_12** load from DLT **DATA.STAR**
3. **SUM_11** physical storage: **HOLAPGP1.SUM_11** load from **HOLAPGP1.SUM_12**
4. **SUM_10** physical storage: **HOLAPGP1.SUM_10** load from **HOLAPGP1.SUM_11**
5. **SUM_8** physical storage: **HOLAPGP1.SUM_8** load from **HOLAPGP1.SUM_11**.
   Note that you cannot load SUM_8 from SUM_10 because SUM_10 is an MDDB and PROC SUMMARY cannot read from MDDB files.
6. **SUM_4** physical storage: **HOLAPGP1.SUM_4** load from **HOLAPGP1.SUM_8**.

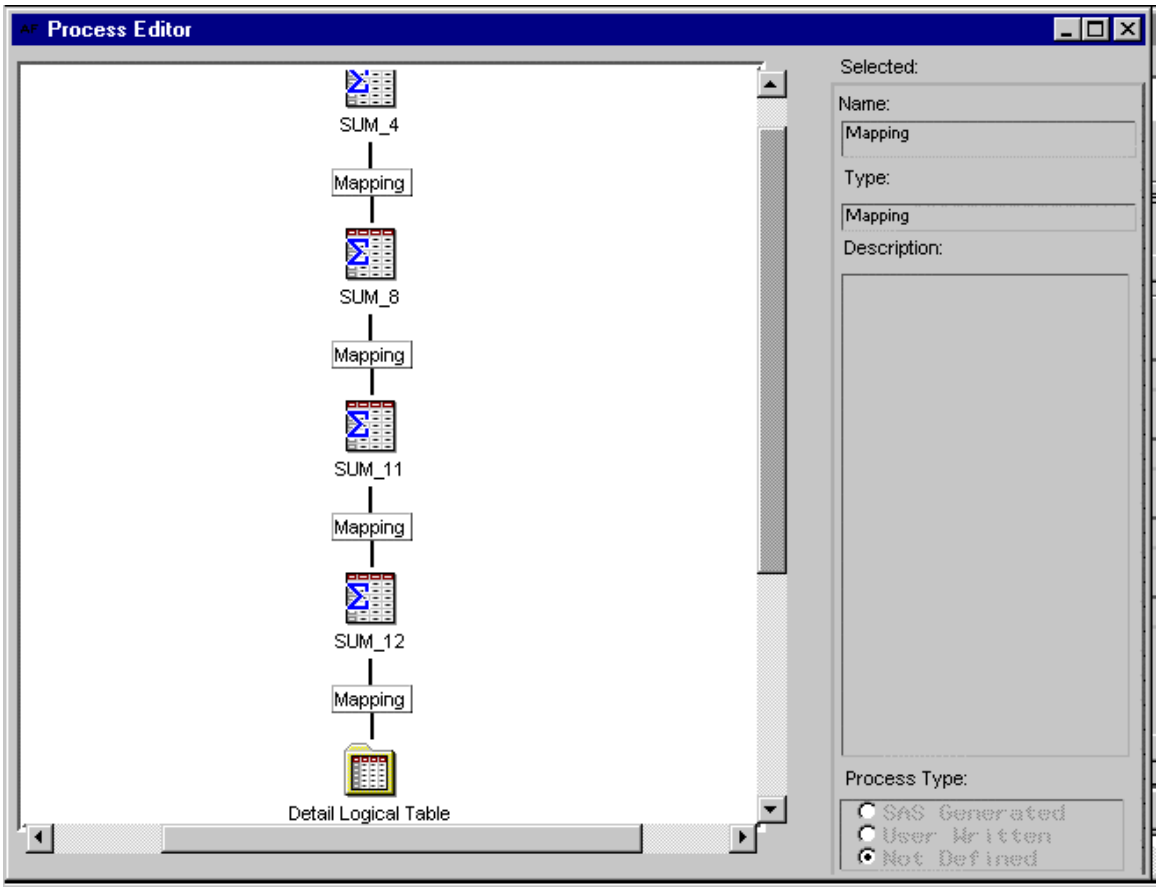


Figure 10: Process editor diagram for SUM_4.

When that process is complete you should be able to use **MB3-Data Utilities-Browse** on each of the items in the group to inspect them.

Export the Metadata to SAS/EIS Software

Following the instructions in the documentation for the OLAP Wizard, you should now create a new SAS/EIS metabase in the OLAP Group library and export the OLAP Group metadata to that metabase. This export process takes less than 3 minutes.

Build HOLAP reports

Using the SAS/EIS Application builder, you can now build multidimensional reports against the PROXY MDDB and view those reports. One of the reports you might want to build would be **GEOGRAPHY** hierarchy down the left side and **PRODUCT** hierarchy across the top. Do not forget when building the report to select the **Advanced...** button and set the Data model to **SASTOOL._DMDDB.HOLAP_M.CLASS** so that the report viewer uses the HOLAP model rather than the MOLAP model. If you fail to do this, the report will show that the proxy has only a single cell in it.

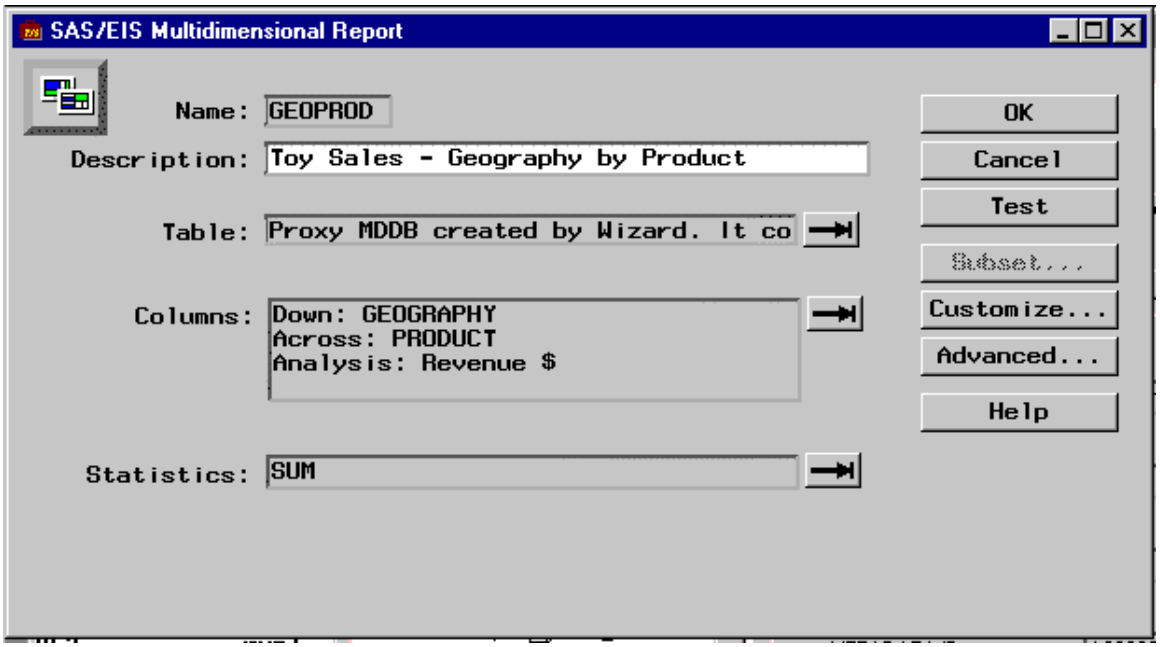
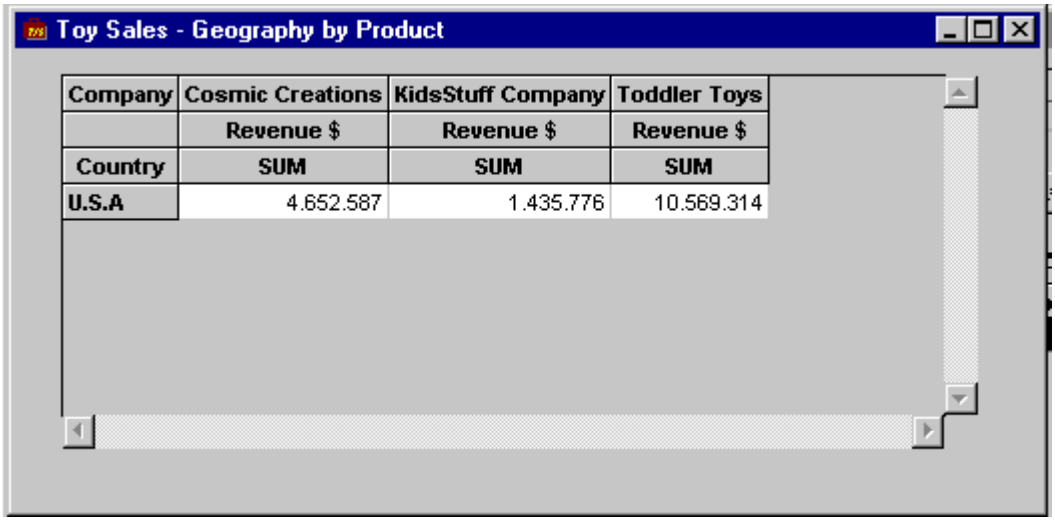
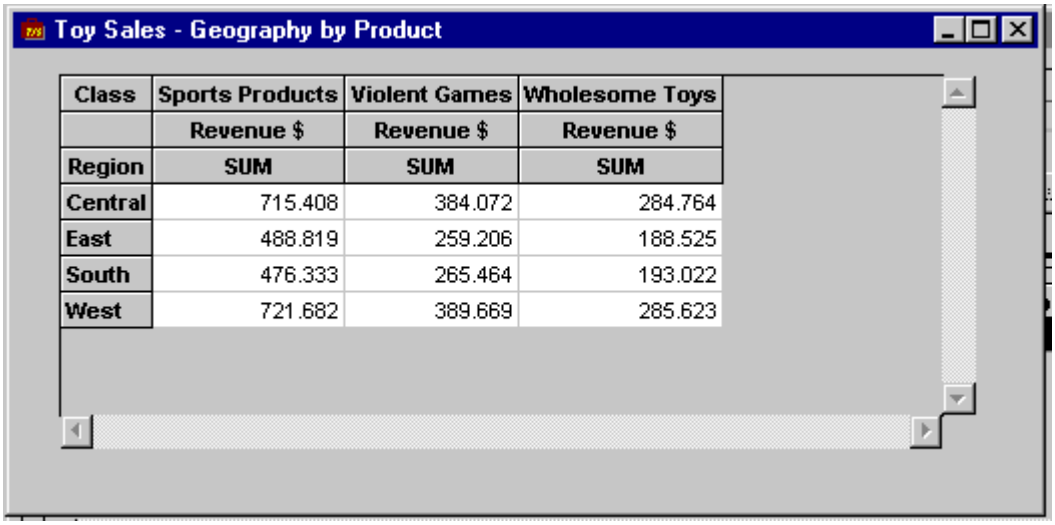


Figure 11: Completed definition of EIS Multidimensional Report.



Company	Cosmic Creations	KidsStuff Company	Toddler Toys
	Revenue $	Revenue $	Revenue $
Country	SUM	SUM	SUM
U.S.A	4.652.587	1.435.776	10.569.314

Figure 12: Initial look of the EIS Multidimensional Report.



Class	Sports Products	Violent Games	Wholesome Toys
	Revenue $	Revenue $	Revenue $
Region	SUM	SUM	SUM
Central	715.408	384.072	284.764
East	488.819	259.206	188.525
South	476.333	265.464	193.022
West	721.682	389.669	285.623

Figure 13: Report after drilling down on both dimensions.

When you are viewing the GEOGRAPHY by PRODUCT report mentioned in the last paragraph, notice the time it takes to drill down from Country to Region. It should be on the order of a few seconds. Notice also the time it then takes to drill down from Company to Product Class. That likewise should take on the order of a few seconds. Notice in the spiral diagram in figure 2 that all four of those class variables are on the spiral and thus in stored crossings. However, when you then drill down one more level in either dimension, it will take several times longer to refresh the display. This is because we did not store the NWAY crossing and the HOLAP model had to go all the way back to the original STAR schema view (DATA.STAR). In doing so, it built a temporary MDDB and displayed that.

Optimizing slower queries

As an exercise, I suggest that you to go back into the OLAP Group and add another crossing that covers the Geographic and Product dimensions at least one level deeper into the spiral diagram. That is, it should include the columns **BRAND** and **STATE** as well as the columns **COUNTRY, REGION, COMPANY, and PCLASS** that are already in the **X0000002** crossing. Note that it does not have to have all of the other columns in **X0000002**. The new crossing could look like this:

```
X0000014 Country Region State Company Pclass Brand
```

You might also consider adding another storage unit as an MDDB called **GEOPROD** to hold that new crossing, although that is not necessary because you could also insert the new crossing into the **SUM_10** MDDB.

To update an **OLAP Group**, simply select it in the Explorer and use **Tools-Add-ins-OLAP Wizard.** Once the wizard displays, you should be able to press **Next** until you come to the **Define stored crossings** page. In that page, simply press **Add** and a new crossing will be added to the end of the list. It will probably be named **X0000014**. At that point you should simply select the six columns **Country, Region, State, Company, Pclass, and Brand** and they should appear in the right side of the window next to the name of the new crossing. Once the new crossing is properly defined, press **Next** to proceed to the **Define storage locations** page. For now, store the new crossing in the existing MDDB. To do this, simply select the **SUM_10** MDDB and the new **X0000014** crossing.

Press **Next** again, then **Finish** on the **Wrap Up** page. Once the wizard updates the metadata with your changes, you will need to **Run** the **SUM_10** MDDB so that the new crossing is added. Finally, re-export the metadata to the SAS/EIS metabase. Once you have updated your OLAP Group to include the new crossing, you should try browsing the report again. This time when you drill down two levels in either the **GEOGRAPHY** or **PRODUCT** dimensions you should see response times improve when you drill-down from **REGION** to **STATE** or from **PCLASS** to **BRAND**.

If you want to try another change to the group you could add another physical MDDB to store the new crossing. First, note the size of the **SUM_10** MDDB that now contains the additional crossing and two new class variables. It should probably be around 23 megabytes. It was less than 4 megabytes before. That is an increase of 19 MB! The addition of the **X0000014** crossing to the **SUM_10** MDDB will improve other queries in addition to the few that you may have tried. This is because it also adds two more variables to the **NWAY** crossing stored in the MDDB. That is why the disk space requirements increase so much. If you just wanted to optimize those specific product and geography queries but did not want to pay the large disk space price then we should place the **X0000014** crossing in an MDDB by itself.

Once again, select the **OLAP Group** and use the **Tools-Add-ins-OLAP Wizard** pull down menu item. Use the **Next** button to skip all the way to the **Define storage locations** page. On that page, select the **SUM_10** MDDB and then click the **X0000014** crossing to cause it to be unselected and therefore no longer included in the **SUM_10** MDDB. To create a new MDDB, simply press **Add** and select **MDDB**. Change the name to **GEOPROD** and select the **X0000014** crossing. That done, finish the wizard. You will now have to setup the physical storage properties and process editor mapping (to the DLT) for the new **GEOPROD** MDDB. Finally, you must **Run** it. You must also re-**Run** the **SUM_10** MDDB in order to rebuild it without the **X0000014** crossing. Finally, re-export the OLAP Group metadata to the SAS/EIS metabase.

When you drill-down into the multidimensional report with this new configuration you probably will not notice any better or worse response times than the previous run because we did not make this change to improve speed but rather to reduce data storage. You should now see that the **SUM_10** MDDB is back to less than 4MB and that the new **GEOPROD** MDDB is only 25 kilobytes! Subtracting that from the 19MB we just trimmed from **SUM_10** we get a net savings of 19 MB with no performance loss. If we compare this solution to the original we see that we improved performance of the third level drill-down quite a lot for a cost of only 25 kilobytes of additional storage.

Conclusion

Starting with the prepared Detail Logical Table, the whole process of creating this **OLAP group** can be completed in less than an hour. Not a single line of code was written. The process of adding a new MDDB or Summary table takes only about 15 minutes. Adding or removing crossings or hierarchies takes about 10 minutes.

If we had first tried the MOLAP solution then decided to change it to the HOLAP plan, the change would have taken about a half hour, saved 44 megabytes of disk space and improved performance of some queries.

In addition to the ease of use of the graphical interface and the ability to quickly tune the warehouse, the OLAP Wizard allows you to build some very powerful OLAP solutions that are easy to maintain and update. This is because it operates within the context of the **SAS/Warehouse Administrator** product. The end result of this work is a fully documented data warehouse that can be updated by regularly scheduled refreshes.

References

1. **OLAP Wizard**
   http://www.sas.com/bin/broker?_program=sps.addtool_61201_sysdep.sas&_service=sps-prod
2. **SAS/Warehouse Administrator 1.3**
   http://www.sas.com/software/components/wadmin.html
3. **SAS/EIS Technical Report: HOLAP Extension, Release 6.12** (SAS publication 56564). Chapter 4 **Implementation Strategies**
   http://www.sas.com/www-bin/bleat@2223/launch/book/56564
4. OLAP Wizard User Documentation.
   http://www.sas.com/software/distribution/html_code/addtool_exp.612-01/holapwiz.html
5. OLAPWizardToyDemo.exe
   ftp://ftp.sas.com/pub/sugi24/AdminOLAPwithSASWA/OLAPWizardToyDemo.exe


```
The following files have been stored on SAS Institute's Internet gateway:

OLAPWizardToyDemo.exe         - self-extracting zip with sample data
ToySpiral.html                - This paper
gifs                          - associated pictures

You can download these files if you have access to the Internet.
To download these files, connect to ftp.sas.com.
Once you are connected, enter the following responses as prompted:

Name (ftp.sas.com: userid): anonymous
Password: your e-mail address

All SUGI 24 files are stored in the following directory:

/pub/SUGI24

There is one subdirectory for each paper that has ancillary files.
For a complete index of all files in /pub/SUGI24, download the following file:

README.index

The file README.index has a description of each subdirectory.
The description will contain the title of the paper and the directory name where the files are stored.
```

This paper is available online by ftp at ftp://ftp.sas.com/pub/sugi24/AdminOLAPwithSASWA
If you can get it in your browser all of these references are hyperlinks.

```
Michael Burns                                  email: sasmkb@sas.com
Principal Systems Developer
SAS Institute, Inc.                            voice: (512)258-5171
PO Box 200075                                  fax: (512)258-3906
Austin, TX 78720-0075
```