Paper 103-2009

Using SAS® OLAP Server for a ROLAP Scenario

Michelle Wilkie, Mary Simmons , Tatyana Petrova, SAS Institute Inc., Cary, NC
John Graas, Teradata, San Diego, CA
Brian Mitchell, Teradata, Miamisburg, OH

ABSTRACT

SAS® offers a flexible online analytical processing (OLAP) solution, not only allowing users to choose a multi-dimensional OLAP (MOLAP) approach, but also a relational OLAP (ROLAP) solution. With the latter approach, the SAS® OLAP Server accesses data stored within relational database management systems directly. A ROLAP approach reduces cube build times and decreases maintenance activities as the data remains in relational database management systems (RDBMS). Using the SAS OLAP Server in tandem with Teradata delivers an ideal OLAP solution for the expanding needs of your organization.

This paper is targeted to advanced SAS users and discusses the following topics:

Relational OLAP (ROLAP) using SAS OLAP Server and Teradata;

Design aspects for a ROLAP implementation;

Teradata performance features and enhancements.

SAS OLAP

Online analytical processing (OLAP) is a technology that is used to create decision support software. OLAP enables application users to quickly analyze information that has been summarized into multidimensional views and hierarchies. By summarizing predicted queries into multidimensional views prior to run time, OLAP tools provide the benefit of increased performance over traditional database access tools. Most of the resource-intensive calculation that is required to summarize the data is done before a query is submitted.

OLAP technology can be further defined by the methods for storing and accessing data. SAS OLAP supports three different variations of OLAP technology:

- MOLAP
- ROLAP
- HOLAP

A SAS cube is comprised of three parts: metadata, navigation files and the physical data or aggregation tables. (See figure 1.) The first two components do not differ with the different OLAP techniques.

The metadata for a cube (created in the SAS Enterprise Intelligence Platform) defines information such as the location of data, the cube structure, cube-based security permissions and calculated measure definitions. The navigation files are used to help understand how the input data information translates to the structure of the cube. For example, how cube members relate to each other, their formats, member properties, and captions for each member.

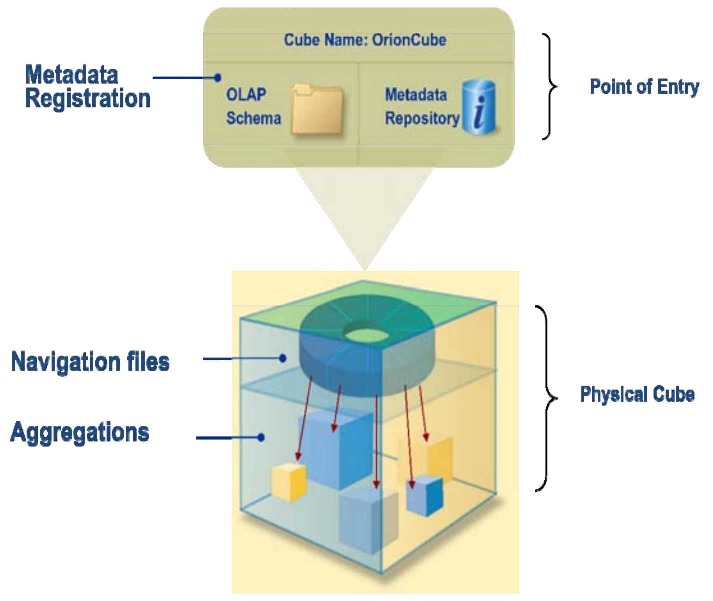


Figure 1. OLAP Cube Structure. The metadata and navigation files components do not differ regardless of OLAP techniques. However, aggregations define the physical data structure, which is unique to each technique.

The physical data is dependent on which OLAP technique the cube designer specifies when building the cube structure. (See figure 2.)

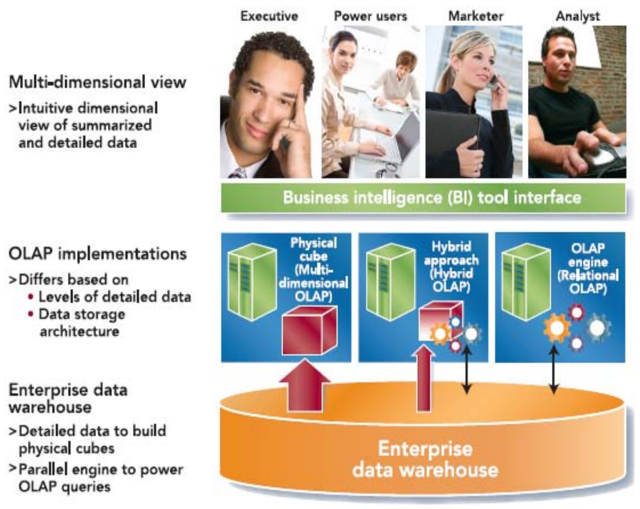

Executive
Power users
Marketer
Analyst
Multi-dimensional view
>Intuitive dimensional view of summarized and detailed data
Business intelligence (BI) tool interface
OLAP implementations
>Differs based on
• Levels of detailed data
• Data storage architecture
Physical cube (Multi-dimensional OLAP)
Hybrid approach (Hybrid OLAP)
OLAP engine (Relational OLAP)
Enterprise data warehouse
>Detailed data to build physical cubes
>Parallel engine to power OLAP queries
Enterprise data warehouse

Figure 2. OLAP Techniques. The Teradata system can optimize OLAP implementations regardless of which technique is used.

- **MOLAP**

With a MOLAP approach, relevant SAS proprietary highly indexed aggregation tables are created and stored within the physical cube. Since the data is pre-aggregated, the response is quickly returned to the end users. While being a popular technique that provides quick access to detail data, MOLAP introduces a set of challenges.

The cost challenge relates to the overhead of the tasks the BI administrators must perform. In a MOLAP structure, the data resides in both the data warehouse and the cube. This means that the data must be updated and maintained in two locations.

When building a MOLAP cube, the majority of time is spent in transferring data and processing aggregations. As businesses expand their analysis to include more dimensions or deeper levels of analytics, the cost and overhead of moving and replicating data into an external cube becomes a challenge for the BI administrator and IT departments.

The common workaround is to reduce the cube size (amount of data to be transferred). However, this results in less data for analysis, fewer dimensions and details, and reduced value for business analysis. Reduction in cube update frequency is also a feasible option but leads to OLAP cubes periodically becoming out of synch with the data warehouse. This can result in less accurate analytics.

- **ROLAP**

To meet the challenges introduced by MOLAP, another technique can be utilized by customers. With a ROLAP approach, all data resides in the relational database management system, where relational tables are optimized for low-level dimensional requests, and aggregate indexes are created for higher-level OLAP requests. A ROLAP-based cube lets the RDBMS handle the SQL and optimization, which may be dependent on implementing an Aggregate Join Indexes (AJI) feature.

The advantages of a ROLAP solution include:

- Only metadata and navigation files are created, resulting in fast build times. ROLAP aggregations build in a fraction of a time it takes to build a MOLAP cube. Data expansion results in minimal impact on cube build time.
- The warehouse updates are immediately available for users querying the cube.
- Data management remains within the RDBMS, not within the cube.

The physical database design of any data warehouse should reflect the customer's business, independent of any tool or application requirements. A normalized data model, snowflake schema, and star schema are all widely used as data models within the enterprises.

SAS OLAP cubes support three types of input data: star schema, detail data, or summarized tables.

Teradata star schema performance has always been sound, and its database has optimizations to support star schema performance. In the case when SAS ROLAP should be implemented on a normalized data model or snowflake schema, an additional semantic layer must be built on top of the table to represent a star schema. Aggregations in the form of Teradata AJIs can then be built on top of the schema to increase OLAP performance.

It is crucial that the normalized model or snowflake schema be "cleansed," meaning there are no NULLs, data transformations are complete, and data is ready for reporting. If not, it may be necessary to build a physical semantic layer, as the Teradata aggregate approach described above will not work on "uncleansed" normalized data. This is also relevant for a star schema design if AJI support is desired.

If a physical semantic layer is required, it is recommended to implement a snowflake schema that is populated by INSERT/SELECTs from the normalized model in the Teradata Database. The INSERT/SELECTs would be defined so that they perform the data-cleansing tasks to result in a snowflake schema that is ready for reporting. A snowflake schema is preferred for Teradata as it yields smaller AJIs (referential Integrity will provide roll-up for higher level columns) versus building an AJI against a star schema, which would require adding the higher levels into the AJI definition.

A SAS cube that feeds from a star schema representation of DBMS data should be defined with the 'Do not create an NWAY' option. This fully summarized table, composed of all crossing of the levels defined in the cube, is equivalent to the PROC OLAP option NO_NWAY.

- **HOLAP**

The HOLAP technique addresses some of the challenges of the MOLAP implementation. With a HOLAP approach, a mix of the SAS proprietary aggregation tables and relational tables is used. This provides the business intelligence administrator the flexibility to establish the location of the multi-dimensional data. The location choice depends on the access frequency, administration, and processing overhead of the data.

The comparative performance and scalability benefits or penalties between the ROLAP, MOLAP, or HOLAP approaches would have to be determined for any given application. A ROLAP solution that uses a star schema or a view built over a snowflake schema, optimized with AJIs, might be the a preferable approach for users who store their data sources on Teradata and utilize SAS OLAP for multidimensional analysis.

DEMONSTRATION DATA

To demonstrate the implementation of ROLAP using SAS OLAP Server and Teradata optimization techniques, we will be using an experimental data sources on Teradata and provide step-by-step guidelines on how to set up your

SAS metadata environment, design cube, and optimize the querying process. For our example, we chose Orion data represented in two scenarios: a star schema and a snowflake schema, as two layouts of data that allow for most beneficial use of Teradata optimization techniques.

Star Schema

Our Star schema consists of a fact table (ORDER_FACT) and four dimension tables (CUSTOMER_DIM, GEOGRAPHY_DIM, TIME_DIM, AND PRODUCT_DIM). (See figure 3.)

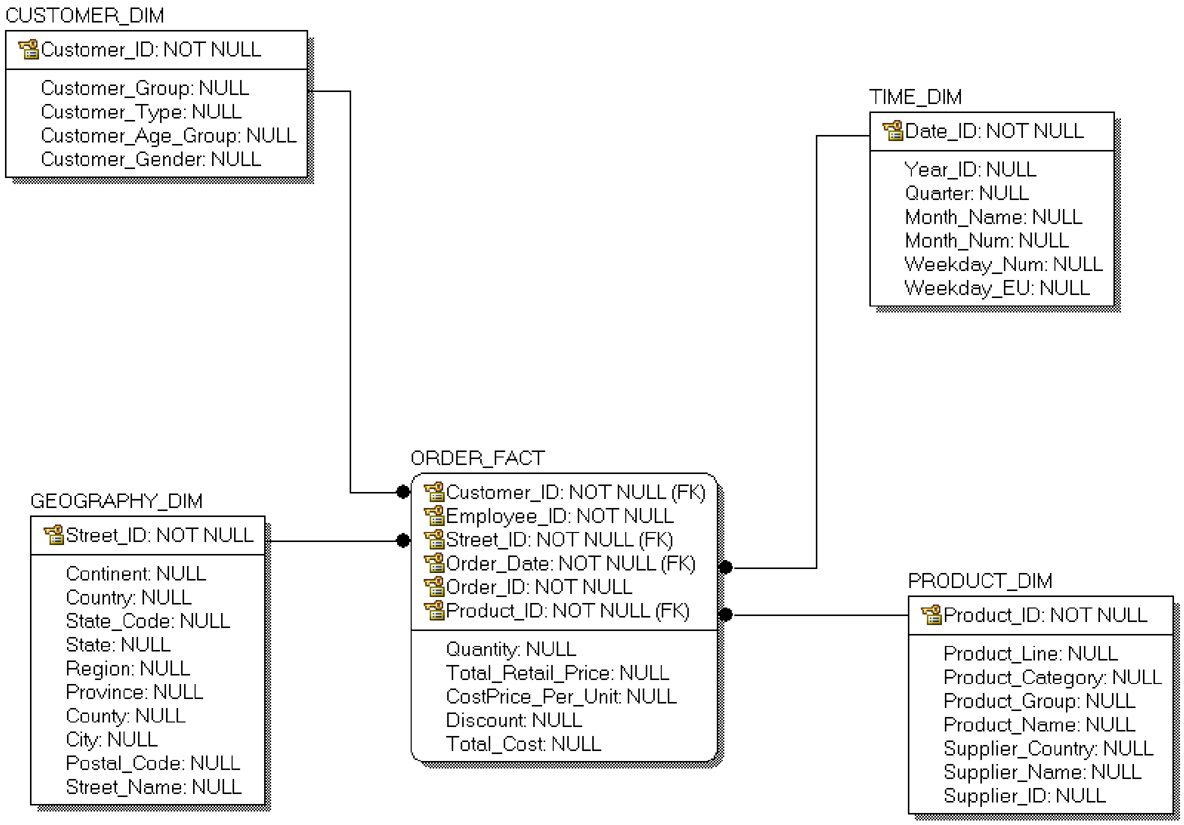


Figure 3. Orion Star Schema.

The physical tables that will not be used for our OLAP cube build are taken out of the picture for simplicity reasons. This is a typical scenario, demonstrating when not all data available would be valuable for the particular cube users and thus are not included into the cube design process. Employee_ID and Order_ID columns from ORDER_FACT table are not referenced from any shown tables. However, they still can be used with corresponding tables in another model if designed for a different business purpose.

Snowflake Schema

Our snowflake model is actually a combination of star and snowflake schemas. We snowflaked data minimally just to demonstrate what considerations should be made when working with a snowflake model rather than a star model (such as adding a semantic layer and other considerations for AJIs set up). For our example, we normalized PRODUCT_DIM table into several tables (productline_dim, productcategory_dim, productgroup_dim, productname_dim, productsnow_dim). To be accessible from SAS OLAP, we added a second semantic layer which consists of a view (productview_dim) on top of normalized tables. (See figure 4.)

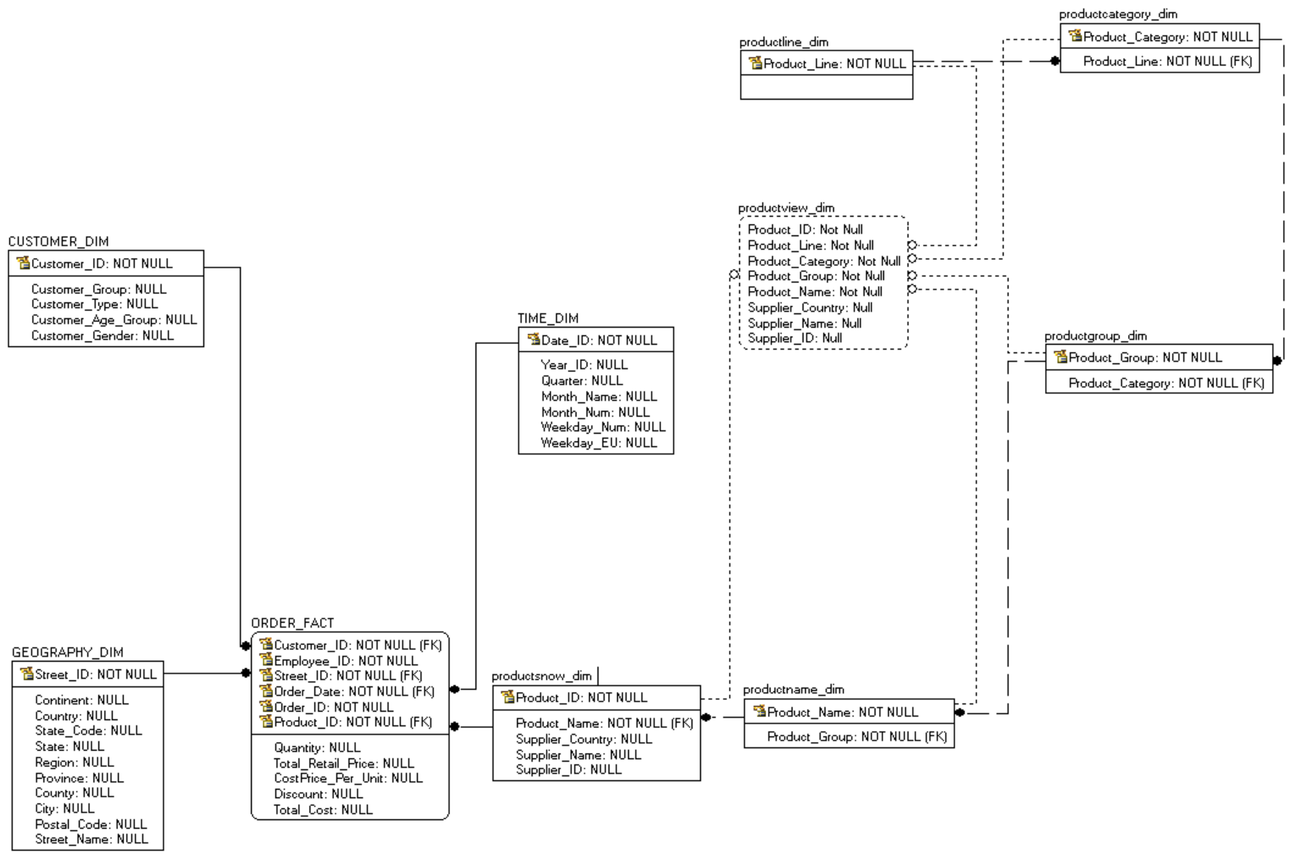


Figure 4. Orion Snowflake Schema.

ROLAP IMPLEMENTATION IN SAS

SAS OLAP SERVER

SAS OLAP uses a combination of SAS servers to store cube metadata, to store the physical cube structure, and to query cubes after they are created. Several types of SAS servers are available to handle different workload types and processing intensities. The SAS OLAP Server is a scalable server that provides multi-user access to data that is stored in SAS OLAP cubes or populated in real-time from relational databases. This server is designed to reduce the load on traditional back-end storage systems by quickly delivering summarized views, irrespective of the amount of data that underlies the summaries. OLAP queries are performed by using the Multidimensional Expressions (MDX) query language in client applications that are connected to the SAS OLAP server.

The SAS OLAP Server has a dual role:

- security validation
    - authentication of the user against the SAS Metadata Server
    - authorization and validation of what the user is allowed to see
- query engine

Which OLAP technique the cube is based on will determine how the MDX query is handled and translated into the appropriate query that will be passed either to an underlying database or internally. In the MOLAP-based cube, the SAS OLAP Server spawns multiple threads internally to retrieve the queries from the relevant cube aggregation tables. For a ROLAP-based cube, MDX is translated into SQL queries, which are passed down to the RDBMS to handle and optimize.

For the most optimized performance at query time, a SAS cube requires aggregation tables that best meet the query result set. SAS provides Application Response Measurement (ARM) logs that help cube designers or administrators tune a cube based on end-user interactions or queries that have been submitted against that cube.

METADATA SET UP FOR A ROLAP CUBE BUILD

A cube designer can define a cube using SAS OLAP Cube Studio or using procedure code (PROC OLAP). To build a cube by using either of those methods, you must complete several preliminary tasks:

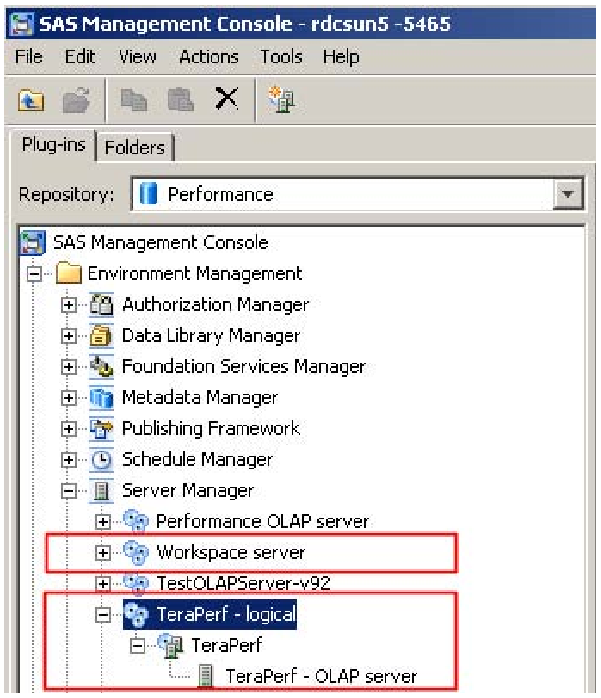
- Define a SAS OLAP Server and OLAP schema
- Define a DBMS server
- Define a SAS library
- Register source data tables

There are different tools available to complete the above steps. For our example we will be using SAS Management Console and Orion data sources on Teradata.

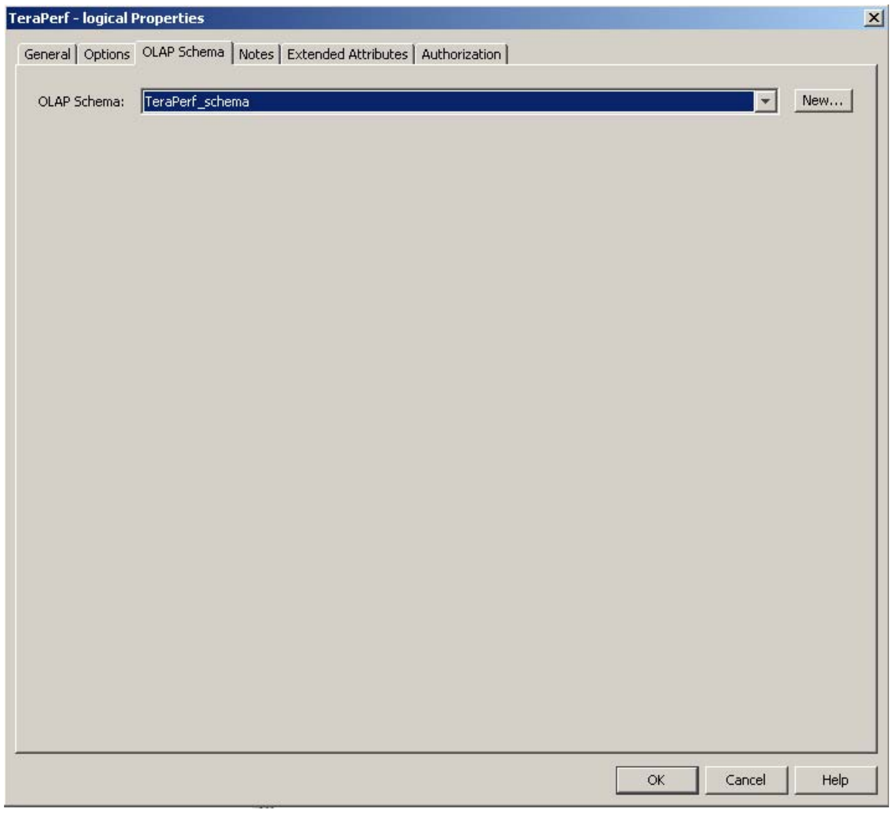
I. Define a SAS OLAP Server and OLAP schema in the metadata.

The server does not need to be running to create cubes, but it must be defined in the metadata. A SAS OLAP Schema is an organization container for SAS OLAP cubes. It communicates with a SAS OLAP Server about which cubes can be accessed by the server and then queried. A SAS OLAP server can be associated with only one OLAP schema. To be able to register data tables later on, you also need to have a SAS Workspace Server registered in the metadata and running while registration takes place.

For our exercise we defined the following servers' structure:



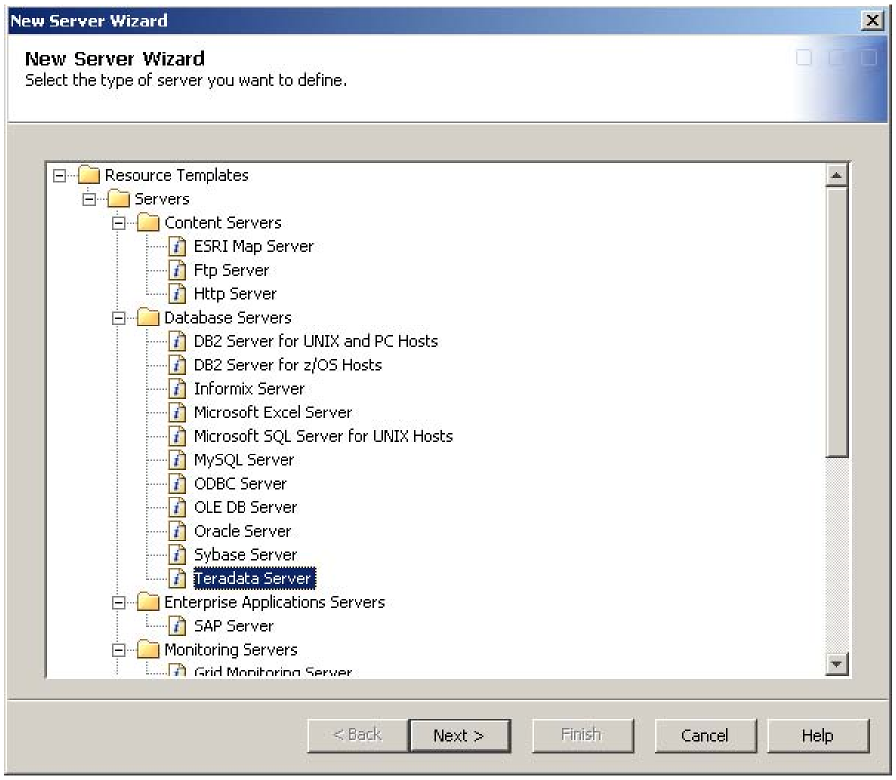
To associate the SAS OLAP Server with the OLAP schema, select the properties of "TeraPerf –logical" server. Select the OLAP schema tab and choose the schema suggested by default or create a new one.



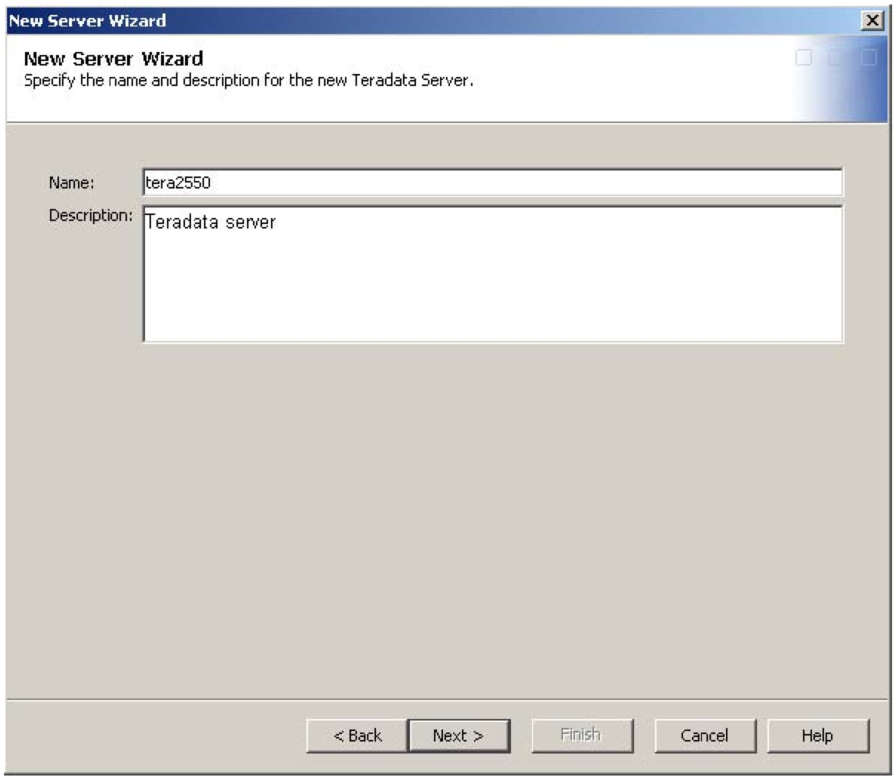
OLAP Schema: TeraPerf_schema

II. Define a DBMS server (in our case Teradata) where the data is stored.

Your library will need to know which data sources to take data from. Defining a DBMS server and assigning it to the library will serve this purpose. To define a DBMS server in SAS Management Console, right click on Server Manager in the Folders Tree and select “New server”. This invokes the New Server Wizard. Specify Teradata Server as the type of your server:



In the next window, specify the name of your server:

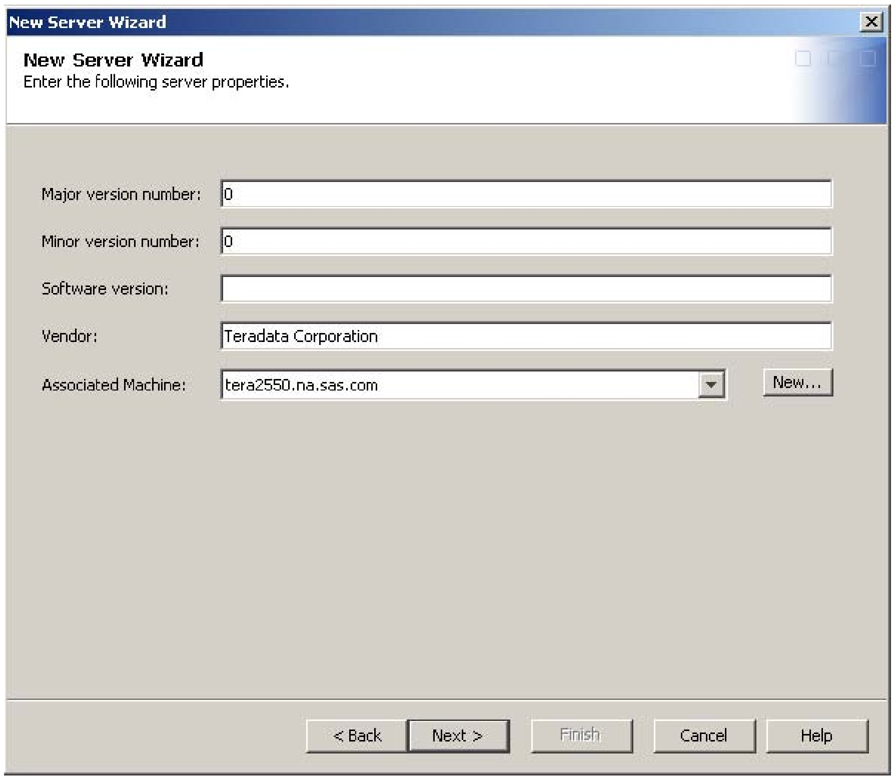


Specify the name and description for the new Teradata Server.

Name: tera2550

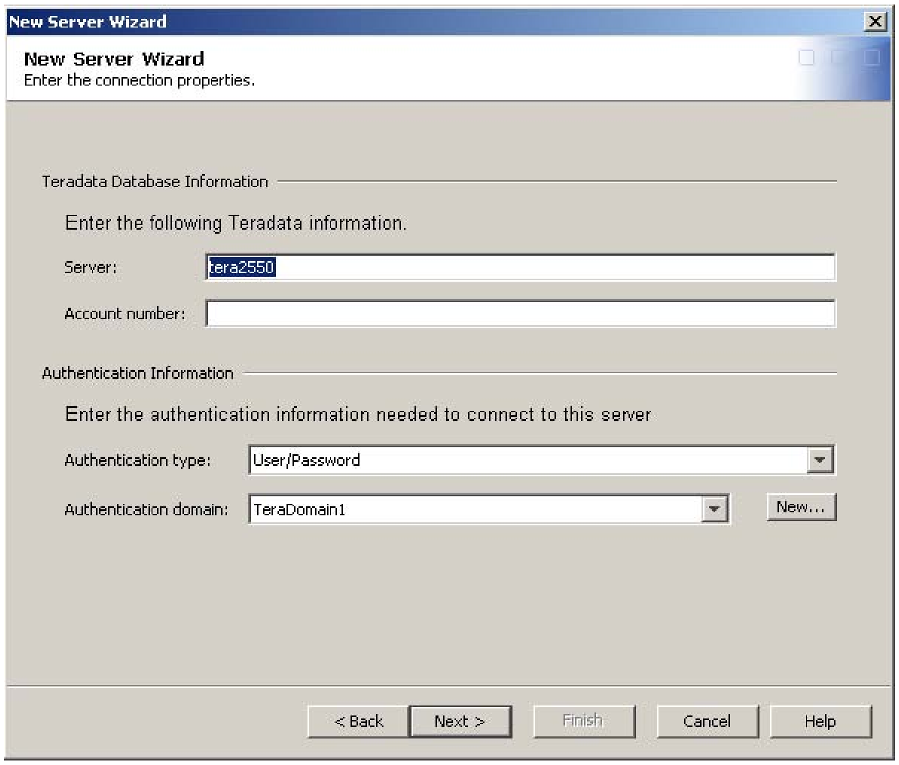
Description: Teradata server

For server properties, specify the machine where your Teradata database is located:



Major version number:	0
Minor version number:	0
Software version:	
Vendor:	Teradata Corporation
Associated Machine:	tera2550.na.sas.com

In the Connection properties window, specify your server name and what type of authentication you need for your OLAP Cube user, as well as which authentication domain will be responsible for passing users' credentials to Teradata.



New Server Wizard
Enter the connection properties.

Teradata Database Information

Enter the following Teradata information.

Server: tera2550

Account number:

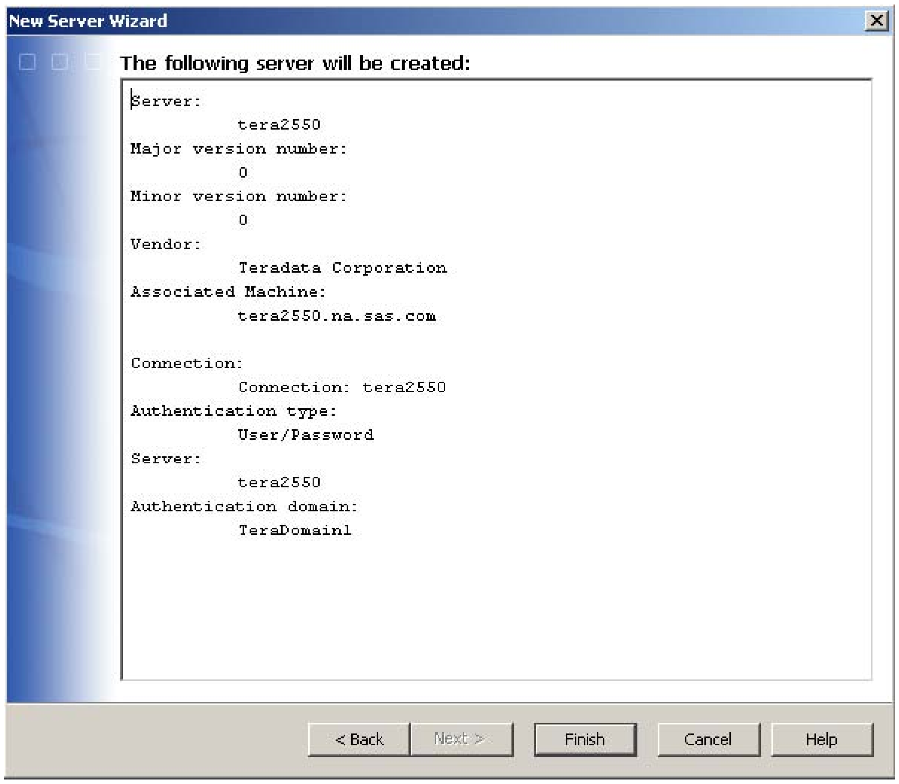
Authentication Information

Enter the authentication information needed to connect to this server

Authentication type: User/Password

Authentication domain: TeraDomain1

Confirm your choices.



The following server will be created:

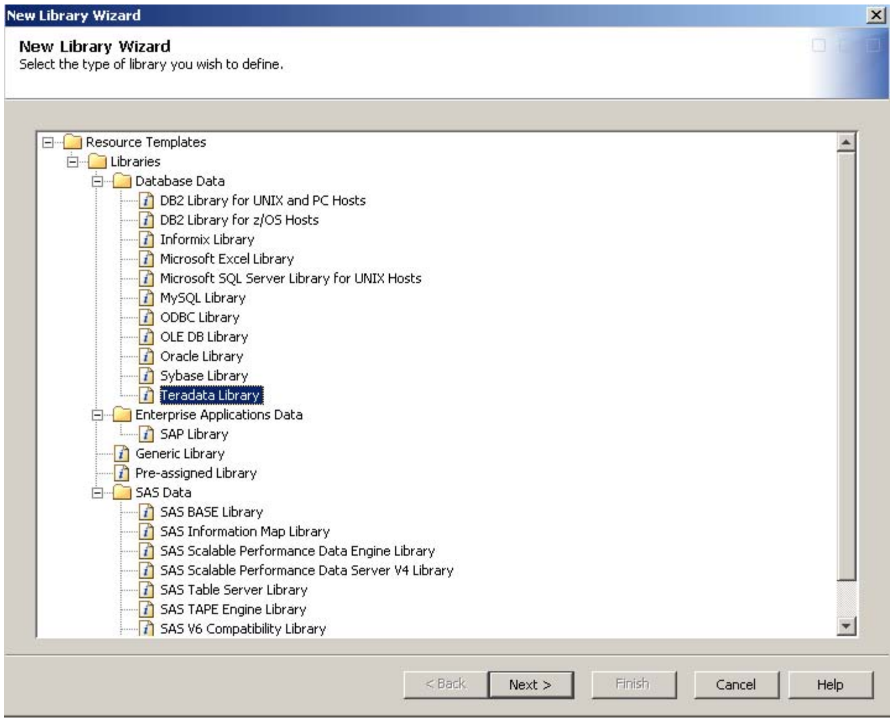
```
Server:
          tera2550
Major version number:
          0
Minor version number:
          0
Vendor:
          Teradata Corporation
Associated Machine:
          tera2550.na.sas.com

Connection:
          Connection: tera2550
Authentication type:
          User/Password
Server:
          tera2550
Authentication domain:
          TeraDomain1
```

III. Define a new library for source data tables

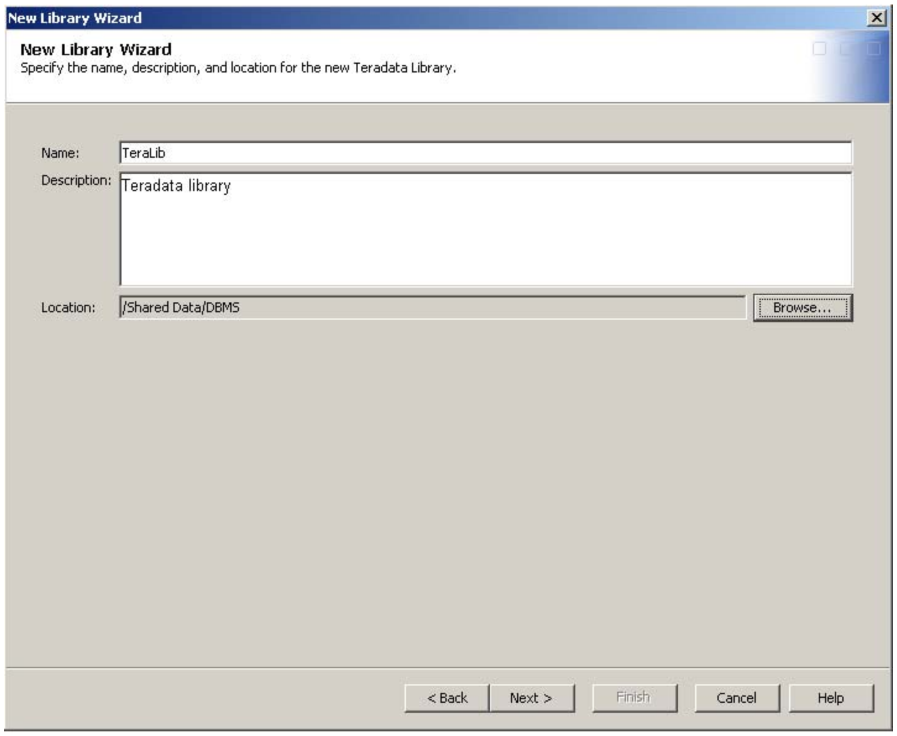
For SAS OLAP, libraries serve as organization containers for the data tables that you use to create SAS OLAP Cubes.

In SAS Management Console, select the Libraries folder under the Data Library Manager in the Folders tree. Right Click on the Libraries folder and select the New Library Wizard. Complete the steps provided by the Wizard as following.

Choose "Teradata Library" as a type of the library:



Name your library and provide the location of the library in the metadata:



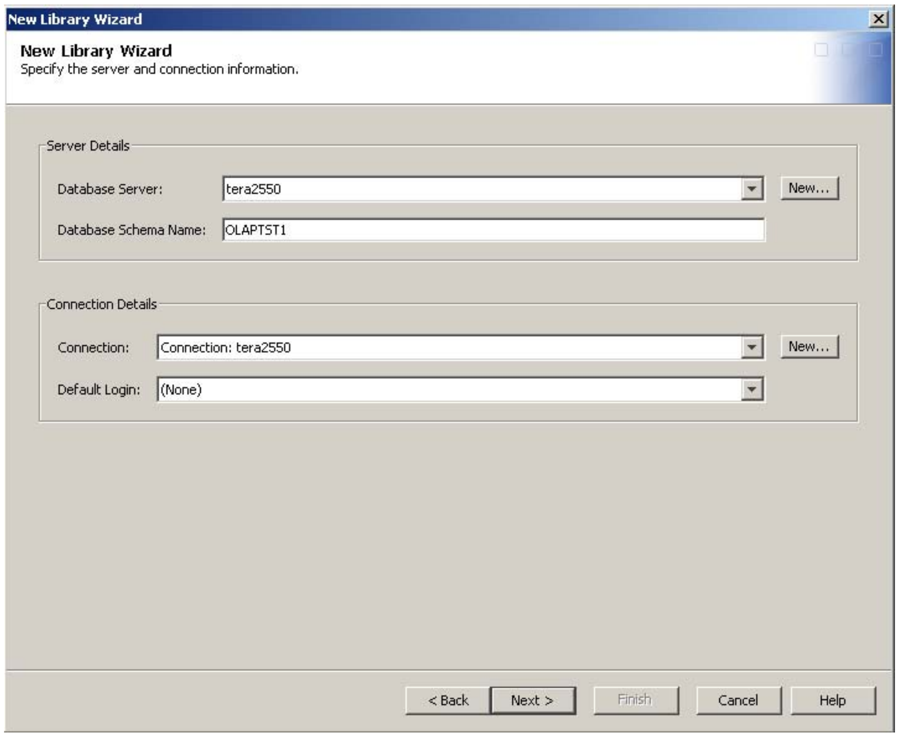
New Library Wizard
Specify the name, description, and location for the new Teradata Library.

Name: TeraLib
Description: Teradata library
Location: /Shared Data/DBMS

On the “Select SAS Servers” window, specify which SAS OLAP server and Workspace server are assigned to the library (picture not shown):

- TeraPerf – logical
- Workspace server

In the next window, specify the database server information:

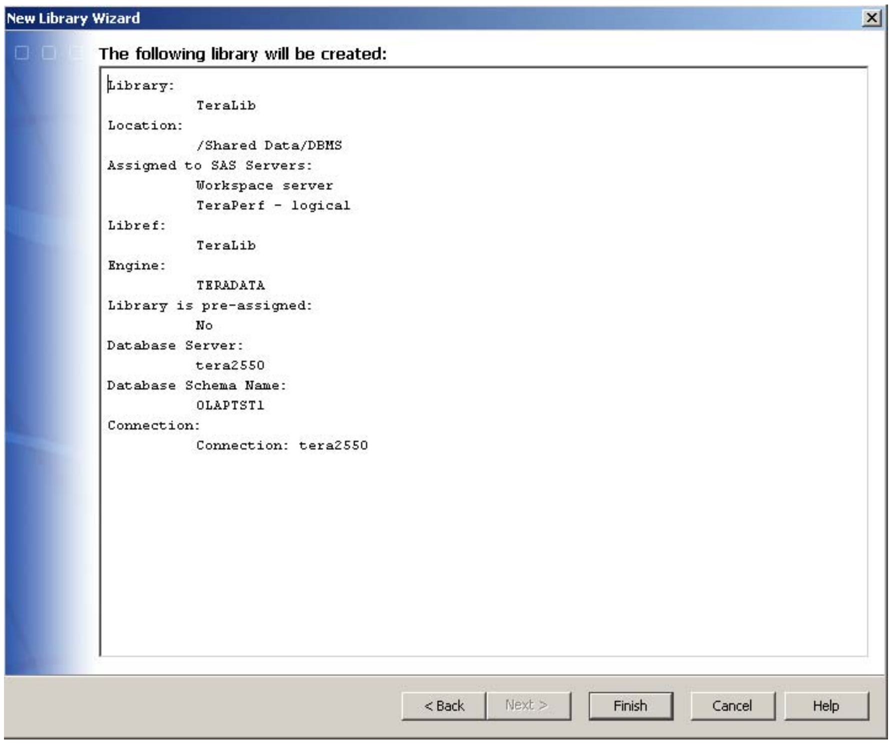


New Library Wizard
Specify the server and connection information.

Server Details
Database Server: tera2550
Database Schema Name: OLAPTST1

Connection Details
Connection: Connection: tera2550
Default Login: (None)

Confirm your choices:

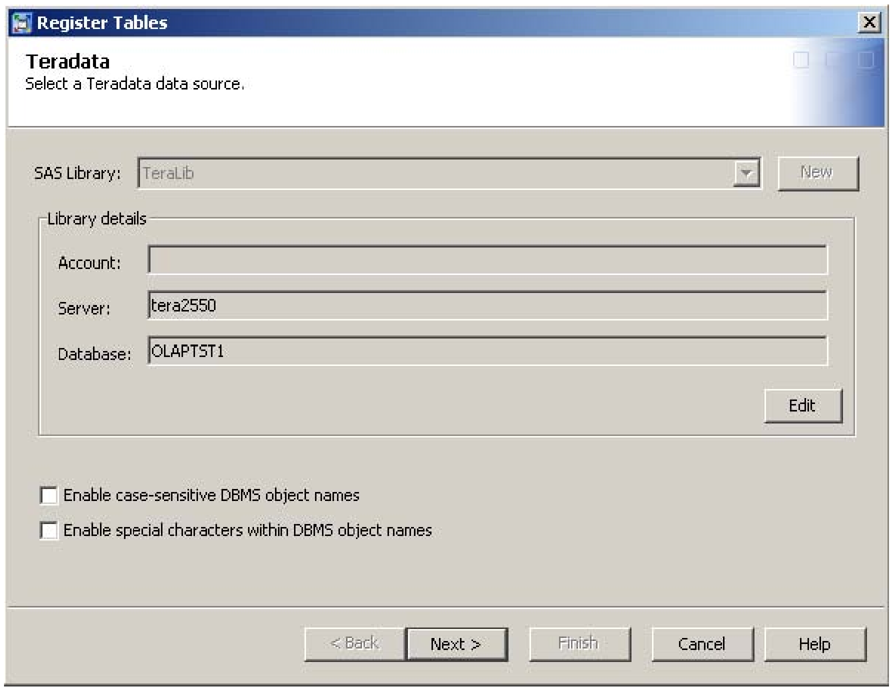


```
Library:
          TeraLib
Location:
          /Shared Data/DBMS
Assigned to SAS Servers:
          Workspace server
          TeraPerf - logical
Libref:
          TeraLib
Engine:
          TERADATA
Library is pre-assigned:
          No
Database Server:
          tera2550
Database Schema Name:
          OLAPTST1
Connection:
          Connection: tera2550
```

IV. Register the source data tables that will be used to create the cube.

For this task we need to register the physical tables from Teradata to the TeraLib library we created above. To do that, Right Click on TeraLib library and select "Register tables". This invokes the Register Tables Wizard. In the Teradata data sources window, the Teradata information is inherited from the Library definition. You have an option here to specify whether to treat your DBMS objects as case-sensitive and whether any of those have special characters in the names.

To be able to succeed with this step, you must have your specified SAS Workspace Server running in the background.


Register Tables

Teradata

Select a Teradata data source.

SAS Library: TeraLib

Library details

Account:

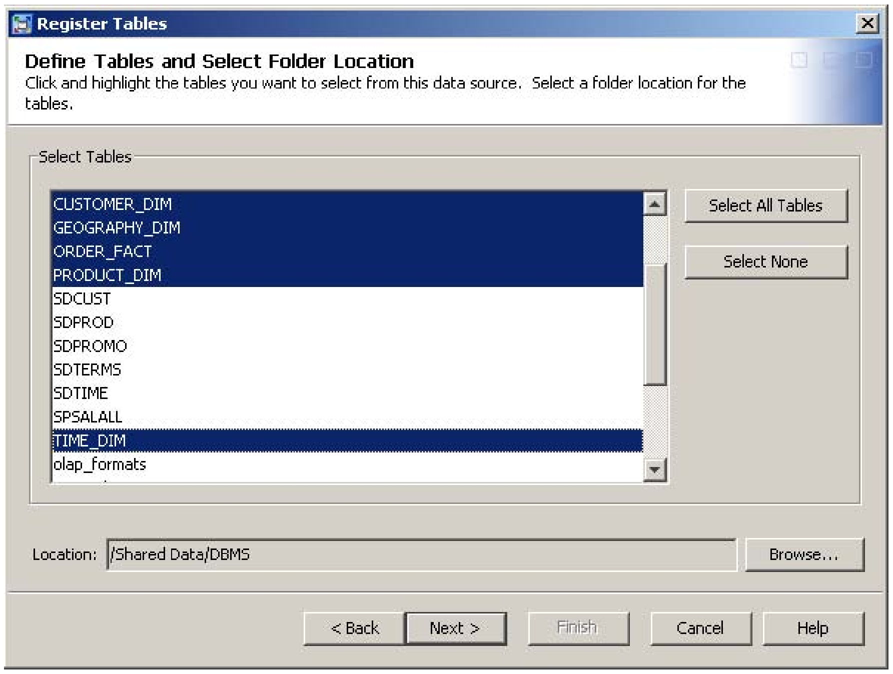
Server: tera2550

Database: OLAPTST1

Enable case-sensitive DBMS object names

Enable special characters within DBMS object names

Select any needed tables from the list of physical tables from the specified database on the specified Teradata server.


Register Tables

Define Tables and Select Folder Location

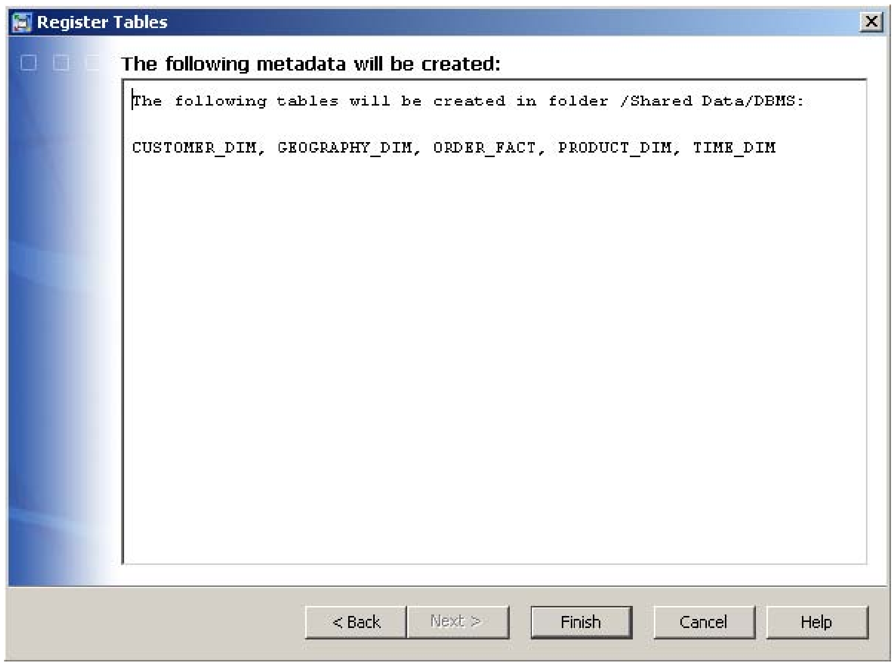
Click and highlight the tables you want to select from this data source. Select a folder location for the tables.

Select Tables

CUSTOMER_DIM
GEOGRAPHY_DIM
ORDER_FACT
PRODUCT_DIM
SDCUST
SDPROD
SDPROMO
SDTERMS
SDTIME
SPSALALL
TIME_DIM
olap_formats

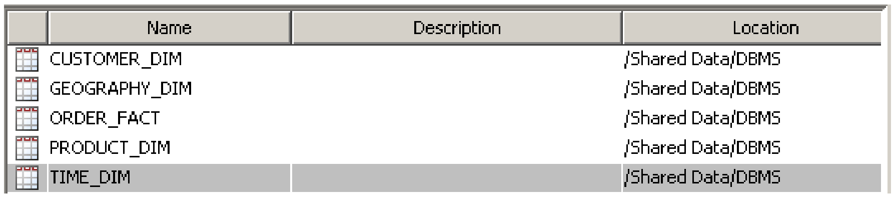
Location: /Shared Data/DBMS

Confirm your choices:



```
The following tables will be created in folder /Shared Data/DBMS:

CUSTOMER_DIM, GEOGRAPHY_DIM, ORDER_FACT, PRODUCT_DIM, TIME_DIM
```

After this task is complete, you can see your registered tables list on the right panel of SAS Management Console when selecting your library:



Name	Description	Location
CUSTOMER_DIM		/Shared Data/DBMS
GEOGRAPHY_DIM		/Shared Data/DBMS
ORDER_FACT		/Shared Data/DBMS
PRODUCT_DIM		/Shared Data/DBMS
TIME_DIM		/Shared Data/DBMS

From this list you can access details of your tables by right clicking on the table and selecting Properties. From there you can view and update details on columns, indexes, keys and authorization information from the SAS OLAP perspective.

Note: Proper authorization permissions should be assigned to the users in order for them to be able to complete the metadata set up and other tasks. This includes the following requirements:

- The user who sets up metadata needs ReadMetadata and WriteMetadata permissions to be assigned.
- The users who will be querying ROLAP cube need to have the set of security permissions assigned based on what data from the cube they are allowed to access. Same users should be defined as members of the Authentication Domain that was specified for your DBMS server.
- Users who will be querying the ROLAP cube should be also defined on DBMS side and have select privileges granted on objects of the physical schema that cube is built upon.

TERADATA GUIDELINES

DESIGN RULES

To provide an optimal ROLAP solution, Teradata recommends the physical database schema to follow a set of design rules which assists proper Aggregate Join Index (AJI) utilization. Follow these design rules:

- All primary and foreign keys are defined as not nullable.
- All primary and foreign keys are not compressible.

- All dimension table primary keys are defined as unique utilizing the UNIQUE constraint, or the primary key is defined as an UNIQUE PRIMARY INDEX.
- Ensure all primary and foreign keys are on ID, not Name or Description columns. This will result in a smaller AJI which means faster access.
- Collect Statistics on all PKs, FKs, PIs, keyword “partition” for all tables with a PPI, and all individual columns included in multi-column indexes for tables and AJIs. This includes Collect Statistics on keyword “partition” for AJIs with a PPI. Also Collect Statistics for all the individual hierarchy columns in denormalized star dimension tables and AJIs.
- Implement referential integrity on the primary key/ foreign key columns. The referential integrity can be defined with the check (hard RI) or no check option (soft RI). The No check option is recommended.

DETAILS OF IMPLEMENTING "DESIGN RULES" ON EXAMPLE

- All primary and foreign keys are defined as not nullable.

  Example: "Street_ID DECIMAL (13,0) NOT NULL,"
- All dimension table primary keys are defined as unique utilizing the UNIQUE constraint, or the primary key is defined as an UNIQUE PRIMARY INDEX.

  Example: "UNIQUE PRIMARY INDEX (Street_ID);"
- Implement referential integrity on the primary key/ foreign key columns. The referential integrity can be defined with the check (hard RI) or no check option (soft RI).

The soft RI option is recommended. There is no maintenance processing cost (think of it as the honor system). Here is an example for the Geography Dimension:

```
ALTER TABLE order_fact ADD FOREIGN KEY (street_id) REFERENCES WITH NO CHECK OPTION
geography_dim (street_id);
```

These rules allow the Teradata optimizer to depend on the data relationships defined and then use the defined AJI for the appropriate access paths. NOT NULL and UNIQUE constraints define the one-to-many relationship versus a possible many-to-many relationship. Referential Integrity constraints allow the elimination of table joins to tables unneeded to resolve the report request. Referential Integrity also enables the optimizer to bring into the query plan (higher level) dimension columns/tables/joins for roll-ups (example: AJI might represent aggregation at the Area level, but with RI in place the Division and Region requests will benefit with the Area level AJI). In addition, statistics allow the Teradata optimizer to aggressively choose the least cost access path to the underlying data.

RECOMMENDATIONS FOR CHOOSING DIMENSION LEVELS FOR AJI

We will demonstrate the example of AJI that was set up for our Orion Star schema. First, we build the dimensional model for all the dimensions that will be used in the cube. (See figure 5.)

Dimension	**Geography**	**Products**	**Time**	**Customers: Hierarchy 1**	**Customers: Hierarchy 2**
	Country	Product Line	Year ID		
	State	Product Category	Quarter		
	Region	Product Group	Month Name	Customer_Type	Customer_Gender
Lowest Aggr	City	Product Name	Week Name	Customer_Group	Customer_Age_Group
Key	Street_ID	Product_ID	Date_ID	Customer_ID	Customer_ID

Figure 5. Dimensional Model.

To support these four dimensions, you can start with creating an AJI at the City/Product Name/Week Name/Customer_Group level. This would be at the lowest level of the hierarchy for each dimension. However, the resulting AJI may be too large, and thus cost more than direct access to the underlying tables for the Teradata database to use for data access. Whether or not an AJI will be used by the SAS report request can be determined by both Explains of the SQL submitted or by DBQL object access counts.

It may be preferable to have an AJI with fewer dimensions included and/or aggregated at a higher level of the dimension hierarchies. Determining the best lowest level AJI will depend on the data demographics and the most common report aggregation levels. For example, an AJI that has a roll-up (group by) at the Region level will support faster queries than one with a group by down at the Street_ID level. The recommendation is to build an initial AJI at the first level up from the fact rows. You can then perform query analysis to confirm that this AJI will be used.

In addition, when a lowest level broad AJI is too large to be used by the Teradata database optimizer, you can use global join indexes for a star schema at the lowest level of the individual dimension hierarchies. An example for Geography dimension is:

```
CREATE JOIN INDEX order_fact_geo1 AS SELECTt street_id, rowid FROM order_fact
PRIMARY INDEX (street_id);
```

When multiple hierarchies are built from the same dimension, and AJI support is required for each of them, (for example Customer dimension) then multiple AJIs should be built. This will depend on the frequency of report requests for each of the hierarchies and whether or not the hierarchies follow the rules of valid primary key/ foreign key relationships.

EXAMPLE OF AJIS SET UP ON DEMONSTRATION DATA

Lowest level broad AJI

```
CREATE JOIN INDEX olaptst1.os_pgc_AJI ,NO FALLBACK AS
SELECT COUNT(*)(FLOAT, NAMED CountStar ),
g.Country ,
g.State,
g.Continent ,
g.City ,
p.Product_Line ,
p.Product_Category ,
p.Product_Group ,
p.Product_Name ,
c.Customer_Group ,
c.Customer_Type ,
c.Customer_Age_Group ,
c.Customer_Gender ,
sum(o.Quantity) (float, named Quantity),
sum(o.Total_Retail_Price) (float, named Total_Retail_Price),
count(o.Quantity) (float, named count_Quantity),
count(o.Total_Retail_Price) (float, named count_Total_Retail_Price)
FROM olaptst1.order_fact o
,olaptst1.geography_dim g
,olaptst1.product_dim p
,olaptst1.customer_dim c
WHERE o.Street_ID = g.Street_ID
and o.product_id = p.product_id
and o.Customer_ID = c.Customer_ID
GROUP BY
g.Country ,
g.State,
g.Continent ,
g.City ,
p.Product_Line ,
p.Product_Category ,
p.Product_Group ,
p.Product_Name ,
c.Customer_Group ,
c.Customer_Type ,
c.Customer_Age_Group ,
c.Customer_Gender
PRIMARY INDEX ( Product_Name, City, Customer_Group);

COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (City);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Customer_Group);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Product_Name);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI INDEX (City,Customer_Group,Product_Name);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Product_Line);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Customer_Group);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Customer_Type);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Product_Category);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Product_Group);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Customer_Age_Group);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Customer_Gender);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Continent);
COLLECT STATISTICS ON olaptst1.os_pgc_AJI COLUMN (Country);
```

Lowest level Product Dimension AJI

```
CREATE JOIN INDEX olaptst1.os_p_AJI ,NO FALLBACK AS
SELECT COUNT(*)(FLOAT, NAMED CountStar ),
p.Product_Line ,
p.Product_Category ,
p.Product_Group ,
p.Product_Name ,
sum(o.Quantity) (float, named Quantity),
sum(o.Total_Retail_Price) (float, named Total_Retail_Price),
count(o.Quantity) (float, named count_Quantity),
count(o.Total_Retail_Price) (float, named count_Total_Retail_Price)
FROM olaptst1.order_fact o
,olaptst1.product_dim p
WHERE o.product_id = p.product_id
GROUP BY
p.Product_Line ,
p.Product_Category ,
p.Product_Group ,
p.Product_Name
PRIMARY INDEX ( Product_Name);

COLLECT STATISTICS ON olaptst1.os_p_AJI INDEX (Product_Name);
COLLECT STATISTICS ON olaptst1.os_p_AJI COLUMN (Product_Line);
COLLECT STATISTICS ON olaptst1.os_p_AJI COLUMN (Product_Category);
COLLECT STATISTICS ON olaptst1.os_p_AJI COLUMN (Product_Group);
```

Here are some additional design rules and observations for the AJI design:

- All queries can use one AJI that joins all the dimensions, even though not all the dimensions were included in the query. That is, the query uses a subset of the AJI dimensions.
- As long as all the columns in the hierarchy (at the level queried or above) are included in the AJI, then the AJI can be used. For example, a query at the region level will use an AJI that has street, city, region, and country columns included.
- Secondary indexes or global join indexes on columns that correspond to low level members in the dimensional model can provide fast access to rows in the FACT table. This will enable the elimination of these columns/values from the AJI, thus making the AJI smaller while still providing access to this level. An example of this might be PRODUCT_ID.

  If the PRODUCT_ID column is normally used in filtering and slicing cubes, and if it is a high cardinality column, then it is a good candidate for a secondary index or global join index.
- The large SQL in-lists being generated by the SAS 9.1.3 can sometimes preclude use of the Primary Index (PI) of the AJI. As of result, the access in these examples might be through a full-AJI scan. This is less prevalent in SAS 9.2.

- To support COUNT, you must have a COUNT summary column in the AJI.
- Ensure all measures in the fact table are set with integer or float data type. Otherwise, overflow may occur on some large calculated measures
- The recommended fact table design is a 'wide' design (columns for each dimension and measure).
- Single level dimensions need supporting reference/lookup/dimension table for optimal performance.
- PPIs allow a table to be partitioned on columns of interest while retaining the traditional use of the PI for data distribution and efficient access when PI values are specified in the query. A typical example for a PPI candidate is the Time dimension, if it is often qualified. PPIs can be used both with the underlying tables and with an AJI.

After AJIs are set up and ready, their usage is automatic by Teradata and transparent for the OLAP users querying the cube. As data grows and evolves over time, and as business analytics requests change, the BI administrator should periodically check whether AJIs in their former set up still provide the optimal optimization.

OTHER HELPFUL TUNING STEPS

Capturing SQL in SAS.

To see the SQL that is passed through, add the following to your autoexec file:

```
   options sastrace=',,,d' sastraceloc=saslog nostsuffix;
```

These captured SQL can then be pasted into the Teradata SQL Assistant tool and then Explained or executed.

DBQL

The Teradata Database Query Log (DBQL) was used to analyze the in-database performance of ROLAP using SAS OLAP server and Teradata optimization techniques. DBQL is the primary data source for query monitoring and evaluation of tuning techniques in the Teradata database. Teradata DBQL provides a series of predefined tables that can store, based on rules you specify, historical records of queries and their duration, performance, and target activity. The DBQL data is accessed with SQL queries written against the DBQL tables or views. DBQL is documented in the Database Administration manual for each release of the Teradata database.

CONCLUSION

SAS and Teradata naturally complement each other with a powerful and flexible solution for BI administrators. Using ROLAP storage in Teradata with SAS OLAP Cubes allows you to reduce data duplication, decrease SAS OLAP cube build times, and keep performance monitoring and optimization tasks within the RDBMS, leveraging already existing knowledge and expertise.

REFERENCES

SAS OLAP Server: Users' Guide

Cube design, by Michelle Wilkie and Arlene Zaima (Teradata Magazine, September 2008)

Implementing AJIs for ROLAP, by Carlos Bouloy (Teradata.com)

An added dimension, by Carlos Bouloy and Rupal Shah (Teradata.com)

RECOMMENDED READING

SAS OLAP Server: Users' Guide

Implementing AJIs for ROLAP, by Carlos Bouloy (Teradata.com)

Teradata Database manuals:

SQL Reference: Statement and Transaction Processing, Chapter 2 "Query Rewrite and Optimization"

Database Management: Database Administration, Chapter 2 "Collecting Optimizer Statistics on Indexes"

Database Management: Database Design, Chapters 9-12

Database Management: Performance Management, Chapter 8 "Collecting Statistics"

CONTACT INFORMATION

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

Michelle Wilkie
SAS Institute
SAS Campus Drive
Cary, NC 27513
E-mail: michelle.wilkie@sas.com
Web: www.sas.com

Mary Simmons
SAS Institute
SAS Campus Drive
Cary, NC 27513
E-mail: mary.simmons@sas.com
Web: www.sas.com

Tatyana Petrova
SAS Institute
SAS Campus Drive
Cary, NC 27513
E-mail: tatyana.petrova@sas.com
Web: www.sas.com

John Graas
Teradata Corporation
17095 Via Del Campo
San Diego, CA 92127
E-mail: john.graas@teradata.com
Web: www.teradata.com

Brian Mitchell
Teradata Corporation
2835 Miami Village Drive
Miamisburg, OH 45342
E-mail: brian.mitchell@teradata.com
Web: www.teradata.com