Improve Your OLAP Environment with Microsoft and Teradata

Rupal Shah, Technical Consultant, Teradata

Richard Tkachuk, Lead Program Manager, Microsoft Corporation

Rev. 1.0

Raising Intelligence

Improve Your OLAP Environment with Microsoft and Teradata

Table of Contents

Scope

This paper describes how you can leverage Microsoft® SQL™ Server Analysis Services and Teradata technologies to improve your analytical OLAP application environment, specifically relational (ROLAP) type solutions. This paper is targeted at Business Intelligence (BI) administrators, system integrators, and database administrators. Readers are expected to have a basic understanding of the features of the Teradata® Database Aggregate Join Index and Microsoft SQL Server 2005 Analysis Services products.

Introduction

Microsoft and Teradata Corporation recognize that customers may want to combine both companies' products when building analytic applications. Microsoft SQL Server 2005 Analysis Services can use data from the Teradata Database in a HOLAP, MOLAP, or ROLAP fashion. Microsoft and Teradata also understand the challenges facing customers today in delivering and accessing important analytic solutions to their end users with access to detailed data. These challenges can range from maintenance and deployment to performance and availability associated with a MOLAP analytic application. This paper is written for those who are interested in building a relational ROLAP solution on a Teradata implementation, which can address these challenges and provide deeper and wider analytics without being overly penalized in performance.

TERADATA
Raising Intelligence

Improve Your OLAP Environment with Microsoft and Teradata

Business Case

MOLAP storage generally provides more rapid query response, but it might not be feasible to regularly move enormous amounts of data to populate the MOLAP store. Teradata and Microsoft have an alternative to address this issue. Defining your cubes/analytics to use ROLAP mode will enable a scalable cube solution resulting in analytics that provide access to an enormous amount of detailed and history data and can be built in a fraction of the time it takes to populate a MOLAP cube from a similarly sized relational source with query performance that meets or exceeds most business requirements.

Challenges

Most of the time it takes to process a MOLAP cube is spent transferring data and populating the MOLAP cache (including aggregates). The data are transferred to a cube building process that resides on a middle server or complex of servers. These challenges are applicable in any analytic implementation as your OLAP environment matures over time for delivering deeper and wider analytics.

Cube Processing Times

As the amount of data stored by the cube increases, through the addition of facts or dimensions, the time required to process the cube increases.

Question: Will my cube(s) build(s) finish in my batch window?

Cube Maintenance

More cubes, more maintenance; larger cube, more hardware resources.

Question: How can I reduce my cube maintenance and hardware?

Network Saturation with Data Transfer

Unless your environment is on a private/dedicated network and/or cubes are built during off-peak hours, data transfer will impact LAN traffic.

Question: How can I reduce network traffic?

Analytic Application to reflect more real-time data

MOLAP cube storage requires processing to make data available to users. Though strategies exist to reduce the amount of data loaded in each process, frequent processing is still required to reduce data latency.

Question: How current are my analytics? Are they current to my data in the warehouse?

Solution

To meet the challenges described, you may be interested in implementing a new solution with direct access to detailed data in the Teradata Database. The solution is:

Analysis Services ROLAP Mode

For users who are interested in extending analytics with a relational solution to reduce cube build times, maintenance, network saturation data transfers, and get real-time detailed data in the warehouse.

Teradata Aggregate Join Index Feature

For users who understand the query performance trade-off that comes with ROLAP (versus MOLAP) storage. Hence, to help attain the best possible performance in this area, we will leverage the Teradata Aggregate Join Index (AJI) feature. Building aggregate join indexes on the Teradata Database eliminates the data transfer and replaces it with high-speed index. These indexes build in a fraction of the time it takes to build and transfer data for a MOLAP cube. Implementing ROLAP cubes with Microsoft and Teradata offers a simple ROLAP solution that enables more dimensions, more history, greater detail, and much faster analytic deployments to end users while still providing MOLAP-like query responses.

An AJI is an aggregated result set saved as an index in the database. It is transparent to end users and BI Administrators, and it will be used automatically by the Teradata Optimizer when a query plan contains frequently made like columns and aggregates. The following sections will cover type of AJI for ROLAP that should be considered. We will assume readers are familiar with this feature. For more information, refer to the Teradata Database SQL Reference – Data Definition Statements.

Improve Your OLAP Environment with Microsoft and Teradata

Note: This paper will not focus on any size or hardware configurations with the Teradata Database or Microsoft products. Since this will vary from customer to customer, the intention here is for you to take away an approach/alternative to improve your OLAP environment.

Environment Considerations

Enterprise Data Warehouse

The enterprise data warehouse (EDW) (physical database design), by definition, should reflect your company's business data independent of any tool or analytic requirement in delivering BI content. An EDW based on Teradata should be designed to adhere to the practices and methodologies to best support an enterprise data warehousing environment for you.

Semantic Layer

To date, customers who have implemented the ROLAP solution have created two semantic layers on top of their EDW to support this solution.

The first is a semantic database layer to hold their analytic specific data content in a star/snowflake schema where Teradata AJI(s) are implemented. Initially, this may be implemented as insert/selects sourced from the EDW into the star/snowflake tables. This activity occurs after the refresh/batch updates are completed on their base EDW tables.

The second is a semantic view layer for the tool/end users to access. This would be a one-to-one (i.e., pointing to the semantic database layer) and include any additional complexities that are appropriate for the analytic to leverage Teradata to do the heavy lifting versus modeling in the tool. The purpose of this layer is to provide an abstract for the users and applications from the physical environment, thus making it possible for changes to be made in table and schema design without disturbing existing reports and applications.

The AJI solution also could have been built off of the customer's 3NF tables, but this would require taking the load processes into account, most likely resulting in a change of load/ETL procedures at the customer site. Plus, specifically with proof of concept (POC) efforts where available resources and timelines are tight, we recommend the above semantic layers for delivering the POC in a timely fashion. See Appendix A for considerations surrounding AJIs on 3NF tables.

Refresh Activities

By defining and building an AJI against a materialized semantic database layer, the warehouse can continue to be updated with minimal impact to end users. At a high level, customers to date have performed the following activities, in one form or another, to refresh various components of their EDW, Semantic Layer, AJI, and Analytic environment:

Refresh Task	Description	Activity	End-User Impact
EDW	Update base tables.	Normal ETL process since no AJI(s) are defined against base tables in this example.	None
Drop AJI	Drop AJI on semantic database layer objects for refresh activities.	DBA activity to DROP AJI object.	Minimal
Semantic Layer	Refresh/update semantic database layer objects.	DBA activity using SQL (i.e., insert/select), Teradata Utilities (e.g., Fastload, Multiload, etc.) or new ETL processes to update objects.	Some
Create AJI	Rebuild AJI against refreshed semantic database layer.	Use Teradata Database parallel processing to build AJI.	Some
Analytic	Refresh analytic to incorporate/ expose new data from warehouse.	BI Administrator to rebuild the ROLAP analytic to reflect new or changed dimensional data.	Minimal

TERADATA
Raising Intelligence

These activities can be viewed as new processes and/or additional resources to consider. These activities on Teradata can update and build in a fraction of the time it would have taken for any extract solution to complete.

OLAP Analytic Design

We will use Microsoft Business Intelligence Development Studio (BIDS) modeling tool to design the Analysis Services dimensional OLAP model from the semantic view layer database.

Proposed Aggregate Join Index

When evaluating an AJI for a ROLAP solution, we propose creating a broad AJI that references all dimensions and levels except the very lowest level(s). Even though our opinion is that this is a reasonable best practice performance, it must be evaluated in each unique customer environment. Determining this cut-point/lowest level of your FACT table in your semantic layer will be imperative to delivering the best analytical OLAP environment to your end users. DBA and BI administrators need to work with the end users to determine the balance between the business requirements (scope of analysis) and service level agreements to provide acceptable performance. Hence, creating an AJI at the lowest level or across every dimension could impact size, time to build, and access, which could be no different than going against the base tables or the total time it takes to build your current MOLAP cube. The intent here is to provide an AJI for those levels and dimensions most used in the OLAP environment.

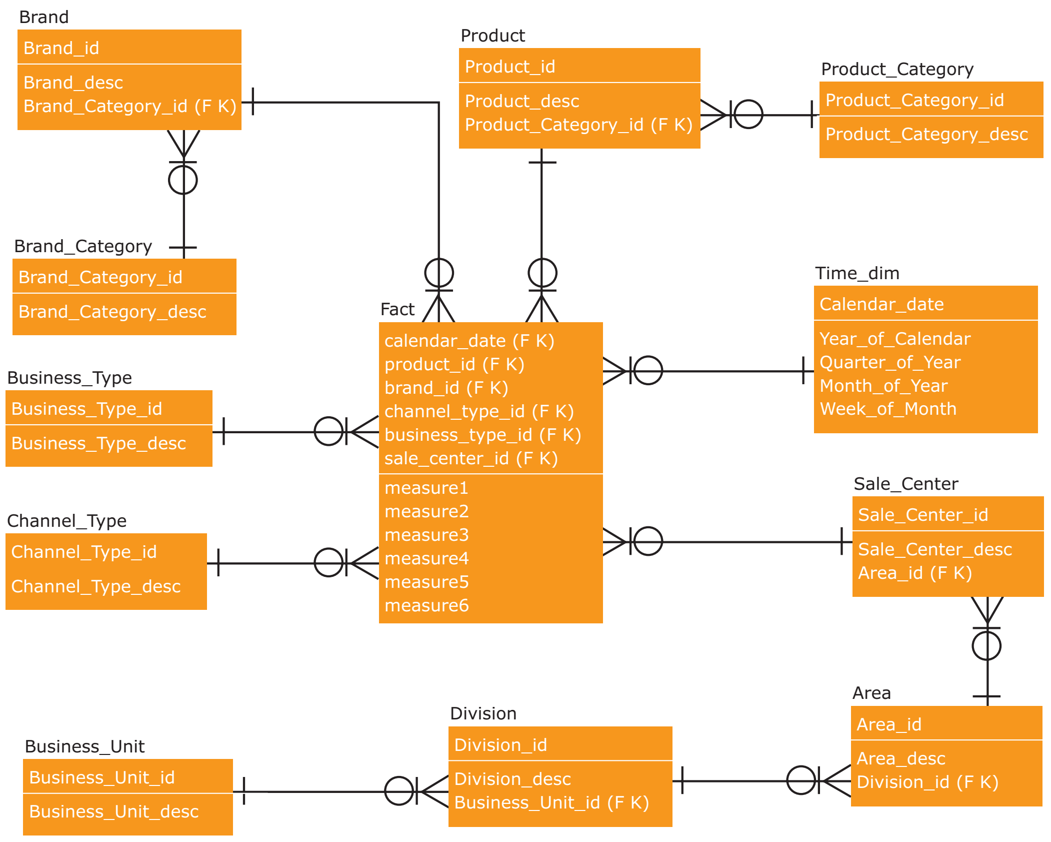


Figure 1.

Dimension Line

Time	Products	Brands	Business	Channel	Org
Year	Product Category	Brand Category	Business Type	Channel Type	Business Unit
Quarter	Product	Brand			Division
Month					Area
Day					Sale Center

Figure 2. Dimension Map

Considerations for Building Aggregate Join Indexes for ROLAP Solutions

The ROLAP solution described in this paper will use this schema/semantic layer database (see Figure 1).

Our analytic looks similar to the dimension map (see Figure 2) from the semantic layer above. It defines all the dimensions and levels that will be accessible in the analytic.

Teradata Database Considerations

To deliver a timely, optimal, and performant ROLAP solution using AJIs, we recommend the following Teradata physical database design to be implemented in a semantic layer database (i.e., analytic specific) as discussed above. Physical table stipulations within this database are:

> Star and snowflake models are currently being recommended. Once partial group bys are enabled on Teradata Database and are supported with AJIs, then a snowflake model is preferred and will yield a more optimal solution.

> 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 a 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.

> Ensure all measures in the Fact Table are set with Integer data type. Otherwise, overflow may occur on some large calculated measures.

> Recommended Fact Table design is 'wide' (i.e., columns for each dimension and measure).

> Single level dimensions need supporting reference/lookup/dimension table for optimal performance.

> Collect statistics on all primary key/foreign key relationship columns.

> Implement Referential Integrity (RI) on the primary key/foreign key columns. RI can be defined with the check (a.k.a., hard RI) or no check option (a.k.a., soft RI).

```
ORGS: Business Unit→Division→Area→Sales Center
ALTER TABLE FACT ADD foreign key
(SALES_CENTER_ID) references with no check option
SALE_CENTER (SALES_CENTER_ID)
```

(Above RI relationship is all that is required for star semantic design (i.e., FACT to one dimension table with lowest level). Otherwise, for snowflake, RI(s) will need to be defined for each higher level roll up where each level is in its own table.)

Improve Your OLAP Environment with Microsoft and Teradata

To address the lowest levels in your analytics (i.e., your FACT table) that are not in your AJI, consider:

Secondary Indexes

Secondary indexes on your FACT columns provide faster set selection. Secondary indexes are frequently selected by the Teradata Optimizer when a search condition cannot be satisfied with primary index retrieval. The optimizer also selects secondary indexes for query plans when they completely or partially cover a query.

Partitioned Primary Index

Take advantage of partitioning for transaction table accesses. A good candidate for OLAP analysis is the DATE column, since it is present in most analytics. MOLAP cubes are challenged in enabling a user to access day level data from within the cube due to the size the cube would be if day were included. Day level data are usually supported via drill-through queries. Partitioned primary index on the date column will enable fast access to day level queries in ROLAP.

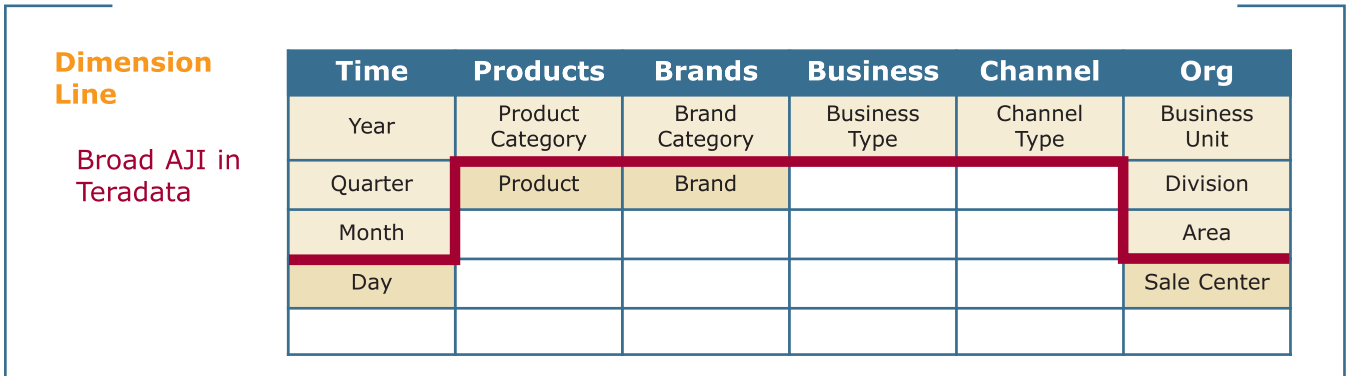
Aggregate Join Indexing Strategy

Determining the columns that will participate in the AJI can be difficult. A good start would be to draw a red line across your dimensional model one level up from the lowest level of each dimension. This is what we call a broad AJI. Single level dimensions, such as Channel Type in this example, are the exception. There is no higher level than Channel Type in the dimension so it should be included in the broad AJI definition (see Figure 3).

> A good rule of thumb for including columns in the AJI definition is to include low cardinality columns in the AJI. High cardinality columns are good candidates for secondary indexes on the fact table. Exclude them from the AJI. High cardinality columns that are defined within the AJI will increase the size of the AJI, thus affecting performance.

> For larger analytics that contain greater than 15 dimensions, it may be necessary to eliminate seldom used dimensions from the AJIs. This will ensure that the highest performance is given to most often used navigations. Seldom used navigations will run slower. Most business users are willing to accept this trade-off given that they are most likely getting more detail, more dimensions, and more timely data with this ROLAP solution.

More experience and better understanding of what type of analyses end users are doing will determine the types of AJI that can be created to improve beyond this initial suggested approach (e.g., an AJI on a specific dimension or AJI across each relational level). This is a DBA task since it will require maintenance (i.e., AJI rebuilds) when the warehouse is refreshed/loaded with new data.



Time	Products	Brands	Business	Channel	Org
Year	Product Category	Brand Category	Business Type	Channel Type	Business Unit
Quarter	Product	Brand			Division
Month					Area
Day					Sale Center

Figure 3.

Aggregate Join Index Definition

Our semantic layer database is a combination of star and snowflake dimensions. This is to show the considerations one needs to understand when defining the AJI (see Notes following). Once the SQL is determined, wrap the CREATE JOIN INDEX and PRIMARY INDEX syntax (see Figure 4) and execute the DML statement via Teradata Queryman or Winddi. Creation time for an AJI will depend on size of the tables and system usage.

Notes:

In our example, to show the various options in selecting an appropriate broad AJI definition our physical design is made of star and snowflake dimensions.

> The TIME dimension is a star single table with a four-level hierarchy. The DDL above includes higher dimension levels (e.g., Year, Quarter, and Month). This is done to ensure the optimizer will use the AJI for higher level queries (e.g., Year) and provide optimal performance in the pure star model or dimension.

> The ORG, PRODUCT, and BRAND dimensions are snowflake multiple tables where multiple tables make up each level of the hierarchy. For example, ORG dimension (four-level hierarchy), higher level roll ups are handled via RI between parent and child tables.

Notice ak.Area_Id reference in DDL is from the lowest dimension (i.e., SALES_CENTER) not from the AREAS table. Hence, unlike star dimensions, you don't need to include higher levels in the AJI definition for the optimizer to rewrite/use the AJI.

However, as mentioned earlier, partial group bys (PGB) are not supported. Once partial group bys are supported with AJIs, snowflake semantic design will be a more optimal solution. This will result in faster and smaller AJI builds, which

```
CREATE JOIN INDEX AJI_Example ,NO FALLBACK ,CHECKSUM = DEFAULT AS
SELECT COUNT(*)(FLOAT, NAMED CountStar ),
          ae.Brand_Category_Id ,
          ac.Product_Category_Id ,
          ad.Business_Type_Id ,
          ad.Channel_Id ,
          ak.Area_Id ,
          al.Year ,
          al.Quarter ,
          al.Month ,
          SUM(ad.Sales )(FLOAT, NAMED SALES )
FROM
          Product  ac ,
          Fact  ad ,
          Brand  ae ,
          Sales_Center  ak ,
          Time  al
WHERE
         (((ad.product_id =  ac.product_id ) AND
         (ad.brand_id =  ae.brand_id )) AND
         (ad.sale_center_id =  ak.sale_center_id )) AND
         (ad.day =  al.day )
GROUP BY ae.Brand_Category_Id, ac.Product_Category_Id, ad.Business_Type_Id,
             ad.Channel_Id, ak.Area_Id, al.Year, al.Quarter, al.Month
PRIMARY INDEX ( Brand_Category_Id, Product_Category_Id, Business_Type_Id,
                      Channel_Id, Area_Id, Year, Quarter, Month );
```

Figure 4.

TERADATA Raising Intelligence

will support any number of levels within a dimension. But, nothing restricts you from building an AJI from a snowflake design today.

> The Channel and Business dimensions are star single table with a one-level hierarchy. As mentioned earlier, single level dimensions, such as these, are the exception. Since there is no higher level in the dimension, it is included in the DDL above. Notice ad.Channel_Id is from the FACT table.

> Notice joins and SELECT clause in DDL is with IDs not with Descs or Names. RI and Teradata Optimizer will take care of SQL requests with Desc or Name. Using IDs will result in a smaller AJI, which means faster access.

> Drop and recreate is the recommendation at this time for rebuilding AJI after warehouse, FACT, and dimension tables have been refreshed. Rebuilding on large installation site will build in a factor of the time versus any cube build.

Check OLAP Query Against the Defined AJI

It's a good idea to always check your relational queries against your defined AJI. Capture the SQL request via Teradata DBQL access logs or from Analysis Services tool logs. Check the request using Teradata Explain command to ensure the AJI is called in the query plan (see Figure 5).

Unless the AJI that was built is very large, it is usually evident from a slow responding query that the AJI is not being used. If the first broad AJI is too large, then another higher level AJI can be built to support higher level queries. Note, the next higher level broad AJI will benefit from the first AJI that was built and will build in a fraction of the time it took to build the first one.

After all indexes are created, there now exist structures to provide fast query performance for a variety of OLAP queries.

> The broad AJI – for most frequently used access paths

> The secondary indexes – on high cardinality Fact column(s)

> The partitioned primary index – on the date column in the Fact table

Explain

```
1) First, we lock a distinct Example."pseudo table" for
   read on a RowHash to prevent global deadlock for
   AJI_EXAMPLE.
2) Next, we lock Example_v.AJI_Example for read.
3) We do an all-AMPs SUM step to aggregate from
   Example_v.AJI_EXAMPLE by way of an all-rows scan
   with no residual conditions, and the grouping identi-
   fier in field 1. Aggregate Intermediate Results are
   computed globally, then placed in Spool 3.  The
   aggregate spool file will not be cached in memory.
   The size of Spool 3 is estimated with low confidence
   to be 2,912,040 rows. The estimated time for this
   step is 8 minutes and 38 seconds.
4) We do an all-AMPs RETRIEVE step from Spool 3
   (Last Use) by way of an all-rows scan into Spool 1
   (group_amps), which is built locally on the AMPs.
   Then we do a SORT to order Spool 1 by the sort key
   in spool field 1. The result spool file will not be
   cached in memory. The size of Spool 1 is estimated
   with low confidence to be 2,912,040 rows. The
   estimated time for this step is 15.92 seconds.
5) Finally, we send out an END TRANSACTION step to
   all AMPs involved in processing the request.
-> The contents of Spool 1 are sent back to the user as
   the result of statement 1.
```

Figure 5.

The only types of queries that are not accounted for in these structures are queries that select low-level dimension members across multiple dimensions without qualifying values. An example of this would be 'Give me the SUM of sales by DAY, by PRODUCT, by SALE CENTERS with no qualifications (WHERE criteria). This business question would result in MANY rows being retuned to the client and would not be considered an OLAP query. The user would most likely be transferring a bulk amount of data back to his PC for analysis using another tool.

Microsoft SQL Server 2005 Analysis Services for ROLAP

This paper is not intended to replace any Microsoft documentation or training, but rather to expose you to the high-level activities with SQL Server BIDS to develop an Analysis Services ROLAP solution against Teradata Database. A majority of this section is excerpts from Microsoft SQL Server 2005 Analysis Services tutorial at http://msdn2.microsoft.com/en-us/library/ms170208.aspx. We encourage you to review this tutorial and be familiar with:

> Dimensional modeling concepts, including facts and dimensions, surrogate keys, and slowly changing dimensions.

> Analysis Services/BIDS fundamentals. At minimum, work through the Analysis Services 2005 tutorial that is available from the start screen of the Analysis Manager or the link above.

> Teradata Database fundamentals, including Teradata physical design techniques, specifically around the proposed solution and concepts mentioned above in the paper.

> Teradata Database open interfaces, such as the .NET Data Provider for Teradata, OLE DB Provider for Teradata, and database features, such as the Aggregate Join Index.

Analysis Services Features Supported by Microsoft and Teradata

Almost every feature of Analysis Services is supported using the Teradata Database. This list includes features that are supported using the Teradata Database:

> Alternative storage modes ROLAP, HOLAP, and MOLAP

> User-defined partitions

> Parent-child dimensions

> Drill-through from the cube to the Teradata Database

> Analysis Services actions

> Data mining

> MDX scripting

Analysis Services Features Not Supported by Microsoft and Teradata

Here is the complete list of Analysis Services features that are not supported by Microsoft and Teradata:

ROLAP aggregations – An Analysis Services cube partition that uses ROLAP partition storage with the Teradata Database must be defined with zero aggregations. This issue is discussed in the Designing Partition Storage and Aggregations section. We recommend using AJI for performance.

Proactive Caching Trace Event Notification Scheme – Trace events are not supported from Teradata when enabling Proactive Caching. However, similar functionality can be enabled with client initiated notifications or polling.

Cell write-back – An Analysis Services analytical application that requires cell write-back cannot be developed directly on a Teradata Database. The cell write-back feature is used most often by budgeting and what-if applications.

Dimension write-back – An Analysis Services analytical application that requires dimension write-back cannot be developed directly on a Teradata Database.

The Microsoft SQL Server Books Online provides an introduction to these features. In particular, the topic Features Supported by the Editions of SQL Server 2005 specifies which of these features is supported in the different editions of SQL Server 2005. Many of the features are available only in the Enterprise and Developer Editions of SQL Server 2000. We highly recommend that you use the Enterprise Edition.

Many aspects of tuning Analysis Services 2005 for performance are common across all data sources. For recommended best practices, please refer to the *Analysis Services 2005 Performance Guide*.

Getting Started

Installation and configuration

The recommended production configuration is to install SQL Server 2005 Analysis Services on a separate server from the Teradata Database. SQL Server 2005 Analysis Services requires the hardware and software indicated in Figure 6.

SQL Server 2005 Analysis Services supports Teradata's OLE DB provider for Teradata and .NET Data Provider for Teradata. Where Analysis Services is installed on 32-bit machines, the OLE DB Provider for Teradata or .NET Data Provider for Teradata is available for connectivity. On 64-bit x64 machines, the 64-bit .NET Data Provider for Teradata Database is recommended. There is no 64-bit x64 OLE DB provider available for Teradata Database, nor are there any plans for one. The .NET Data Provider for Teradata is the advocated interface for Teradata Database. Readers should review Appendix B – General Teradata Considerations (under Connectivity section) for installation and implementation notes.

Analysis Services in BIDS

BIDS is the environment that you will use to develop Online Analytical Processing (OLAP) analytics in SQL Server 2005 Analysis Services (SSAS). BIDS is the Microsoft Visual Studio 2005 environment with enhancements that are specific to business intelligence solutions. For more information about the general features of BIDS, see *Introducing Business Intelligence Development Studio*.

Notes:

> Analysis Services, BIDS, and SQL Server Management Studio can be used out of the box without any special configurations to access a Teradata Database system, if .NET Data Provider for Teradata has been installed and configured properly, see Appendix B – General Teradata Considerations.

> BIDS is a modeling component that application developers can use to build analytics for Analysis Services and other Microsoft applications (e.g., Reporting Services and Integration Services).

Hardware and Software Requirements	AS defined in the Books Online topic Hardware and Software Requirements for SQL Server 2005.
Analysis Services	Microsoft SQL Server 2005 Analysis Services (Service Pack 2) or higher. Editions include: > Standard Edition > Enterprise Edition contains a superset of features from the Standard Edition > Developer Edition is feature-equivalent to Enterprise Edition Many customers use Developer Edition on their development servers. If you are using Standard Edition in the production environment, be aware of the features available in Developer Edition that are not available in Standard Edition. These are detailed in the Books Online topic Features Supported by the Editions of SQL Server 2005.
Connectivity	> OLE DB Provider for Teradata version 1.3 or higher > .NET Data Provider for Teradata version 1.1 or higher.*
Teradata	V2R6.0.1 or higher

*The .NET Data Provider requires changes made subsequent to SQL Server 2005 SP2 not yet available at the time of publication. Please contact a Microsoft support representative for more information.

Figure 6.

> System Management Studio (SSMS) is used by IT Professionals to manage business intelligence applications in production.
> Analysis Services (AS) is a multi-dimensional database that can support various flavors of OLAP (i.e., MOLAP, HOLAP, and ROLAP) applications. It can be built from a variety of sources including an RDBMS with drill-down and drill-through capabilities back to originating source.

The BIDS component provides a way to build and manage a reusable analytic framework to deliver analytics to your end users. Once built, Analysis Services for ROLAP provides seamless access to your relational database for your end-user navigation and analysis (see Figure 7).

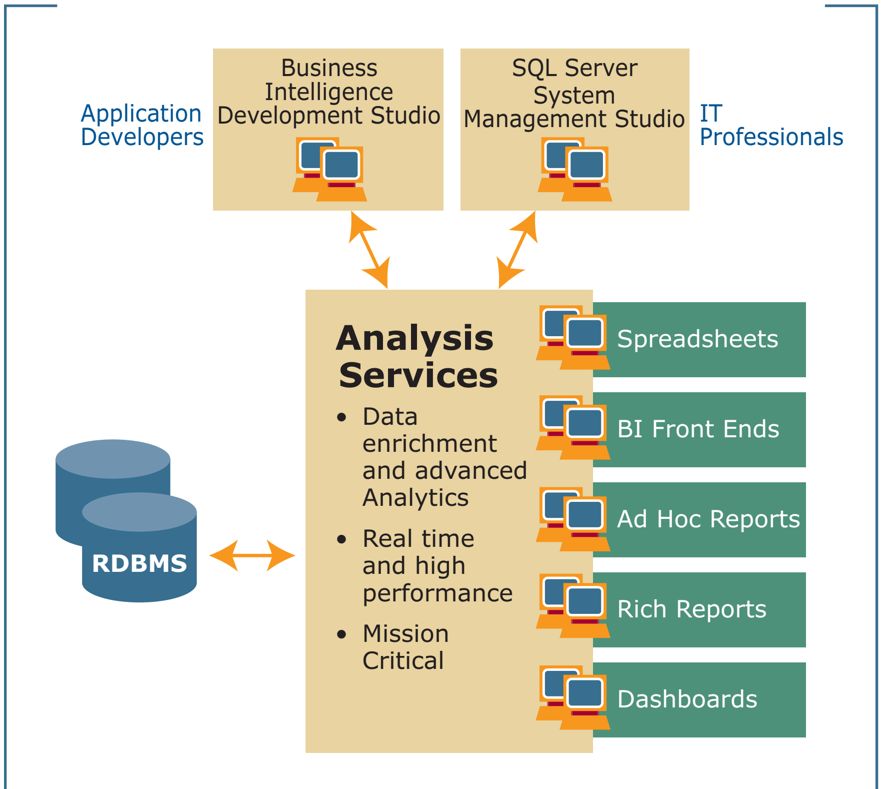

Application Developers
Business Intelligence Development Studio
SQL Server System Management Studio
IT Professionals
Analysis Services
• Data enrichment and advanced Analytics
• Real time and high performance
• Mission Critical
RDBMS
Spreadsheets
BI Front Ends
Ad Hoc Reports
Rich Reports
Dashboards

Figure 7. Analysis Services components.

Design Considerations

For the most part, the steps outlined here and the logical design of an OLAP analytic that is defined against a Teradata Database will be the same as that for any other source with the exception of the unsupported features, described in Appendix C, which do not have any impact when designing a ROLAP analytic. However, the other considerations mentioned in Appendix C need to be in place for optimal relational behavior. Hence, the activities listed for this solution will rely on the semantic layer/schema identified above, modeling/designing the analytic in BIDS, setting the Analysis Service storage mode for ROLAP, and creating a broad AJI.

Start BIDS Tool

Designing a business intelligence application starts with creating a Microsoft project in BIDS. Within this project, you define all the elements of your solution. BIDS opens after you create a new Analysis Services project by using the **New Project** dialog box in Visual Studio 2005. Analysis Services projects and other Business Intelligence project types are available once SQL Server 2005 client tools are installed. BIDS includes a set of windows for all phases of solution development and project management. For example, BIDS includes windows that let you manage multiple projects as a unit and view and modify the properties of objects in projects. These windows are available to all the project types in BIDS.

Improve Your OLAP Environment with Microsoft and Teradata

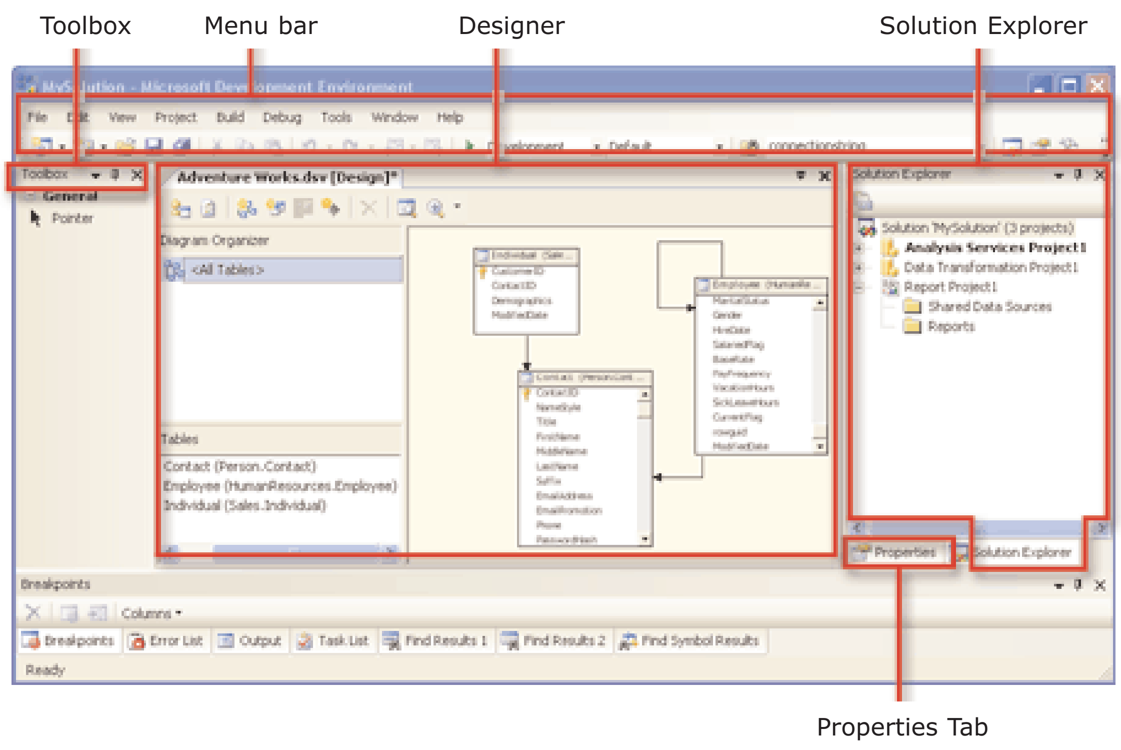

Toolbox
Menu bar
Designer
Solution Explorer
Properties Tab
MySolution - Microsoft Development Environment
File Edit View Project Build Debug Tools Window Help
Toolbox
General
Pointer
Adventure Works.dsv [Design]*
Diagram Organizer
<All Tables>
Tables
Contact (Person.Contact)
Employee (HumanResources.Employee)
Individual (Sales.Individual)
Solution Explorer
Solution 'MySolution' (3 projects)
Analysis Services Project1
Data Transformation Project1
Report Project1
Shared Data Sources
Reports
Properties
Solution Explorer
Breakpoints
Columns
Breakpoints Error List Output Task List Find Results 1 Find Results 2 Find Symbol Results
Ready

Figure 8.

Figure 8 shows the windows in BIDS in the default configuration.

BIDS consists of four main windows:

> Solution Explorer

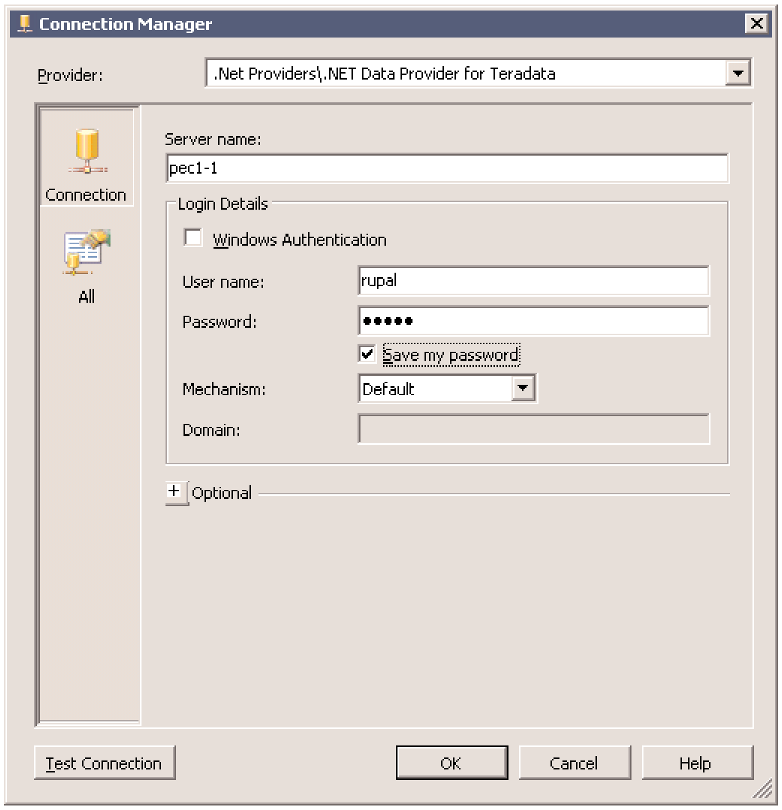
> Properties Window

> Designer Window

> Toolbox

Other windows included in BIDS let you view search results and get information about error messages and information that are output by the project debuggers or designers. Server Explorer lists database connections. Object Browser displays the symbols available to use in a project. Task List lists user-defined programming tasks. Error List provides detailed descriptions of errors.

TERADATA
Raising Intelligence



Provider: .Net Providers\.NET Data Provider for Teradata

Server name: pec1-1

Login Details

Windows Authentication [ ]

User name: rupal

Password: ●●●●●

Save my password [✓]

Mechanism: Default

Domain:

Figure 9.

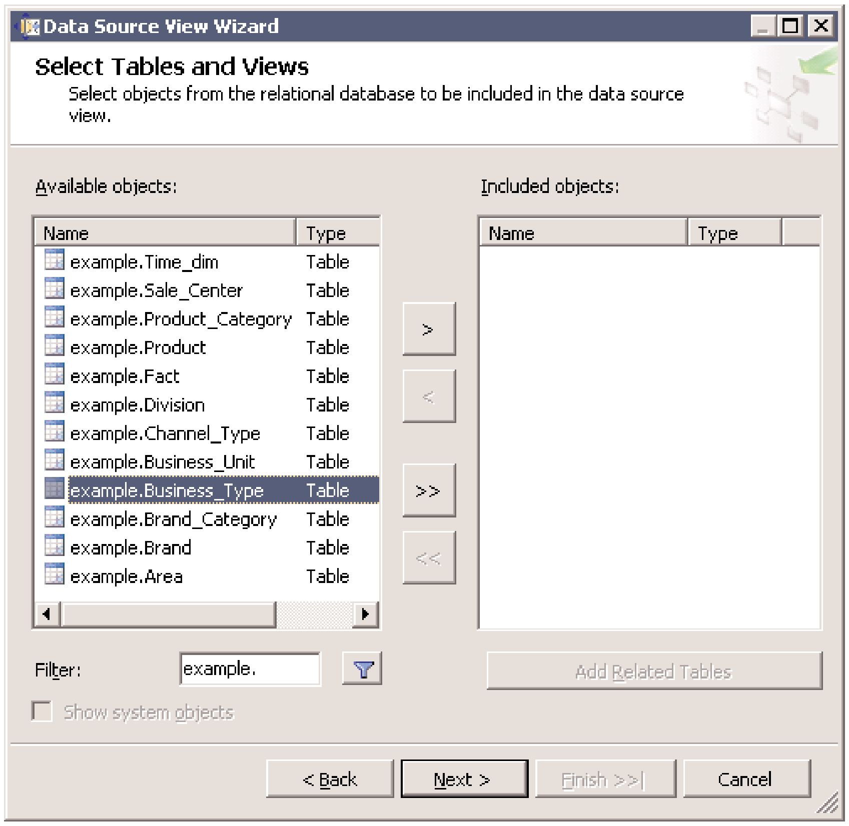
Defining a Data Source Using the Data Source Wizard

The first step is to define a Data Source. Within Solution Explorer, you use the Data Source Wizard in BIDS to define one or more data sources for a Microsoft project. SQL Server 2005 Analysis Services supports many different types of providers, the OLE DB and .NET Data Providers for Teradata being among them. After you select a provider, provide specific connection information required by that provider to connect to the underlying data. The exact information required depends on the provider selected, but generally for Teradata, the information includes the database server name, information for logging on to the database server, and other provider-specific settings.

Figure 9 shows the window for Connection Manager and available provider settings.

Tip: Test Connection for connection status.

See Appendix B – General Teradata Considerations (under Connectivity section) for specific provider settings and recommendations.


Data Source View Wizard
Select Tables and Views
Select objects from the relational database to be included in the data source view.
Available objects:
Included objects:
>

Name	Type
example.Time_dim	Table
example.Sale_Center	Table
example.Product_Category	Table
example.Product	Table
example.Fact	Table
example.Division	Table
example.Channel_Type	Table
example.Business_Unit	Table
example.Business_Type	Table
example.Brand_Category	Table
example.Brand	Table
example.Area	Table

Filter: example.

Figure 10.

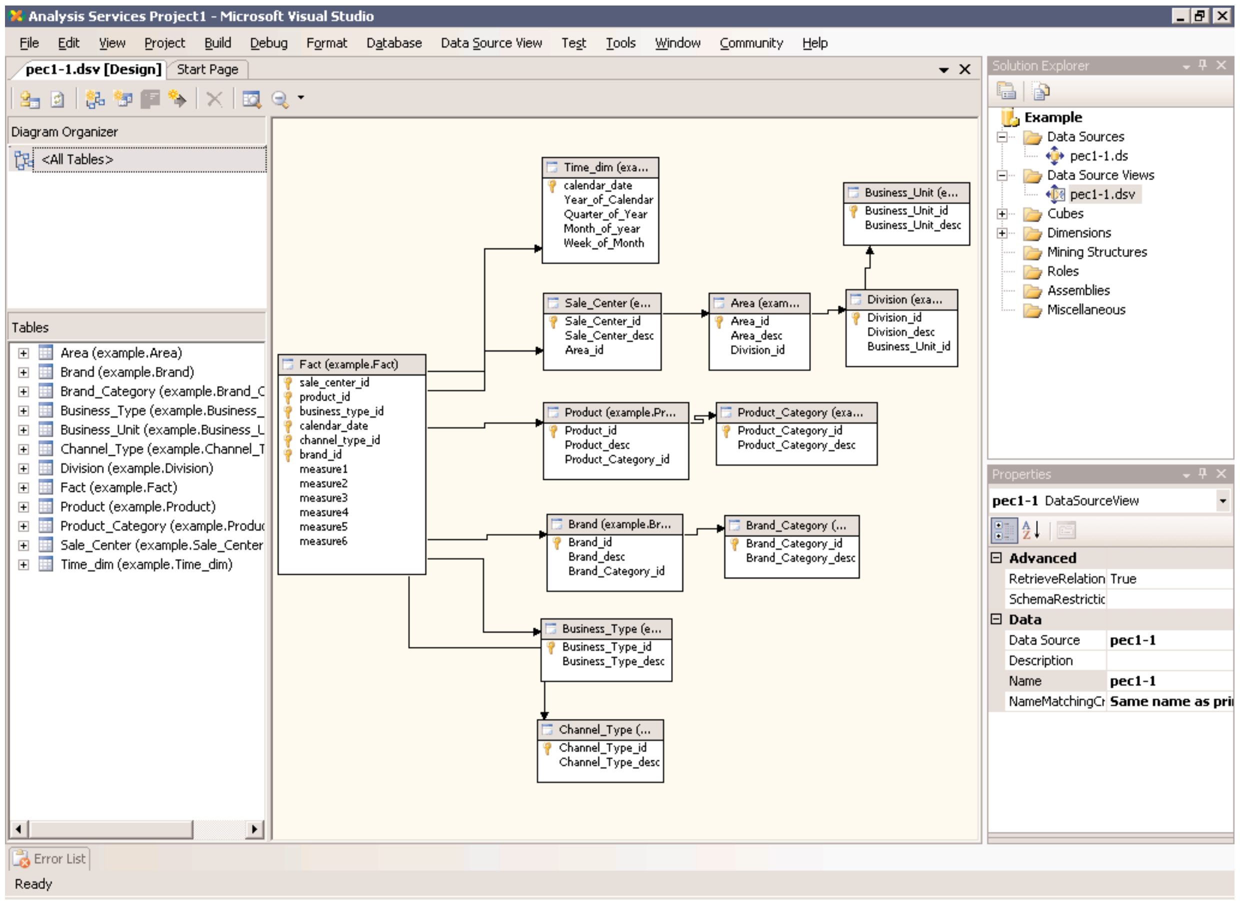
Defining a Data Source View Using the Data Source View Wizard

Once you've defined your Data Source, within Solution Explorer you use the Data Source View (DSV) Wizard in BIDS to define a new data source view in an Analysis Services project based on a data source (e.g., reference database objects views and/or tables that are required for the Analysis Services analytic you are designing).

All objects are listed in the DSV to which the user has access. To limit the list of tables to a particular schema, filtering option on a schema is available as shown in Figure 10.

When you create a data source view, relationships are created between tables/objects based on foreign key constraints in the data source. However, you may define or alter the automatically created primary key keys and relationships within the DSV. Either way, the relationship between fact and dimension objects must be defined within the DSV.

Casting of column/data into different data types can be done either in the source database (e.g., as part of a view) or in the DSV using a Named Calculation which is, essentially, a derived column defined in the DSV. However, care should be taken when using



pec1-1 DataSourceView

Advanced	
RetrieveRelation	True
SchemaRestrictic	
Data	
Data Source	**pec1-1**
Description	
Name	**pec1-1**
NameMatchingCr	**Same name as pri**

Figure 11.

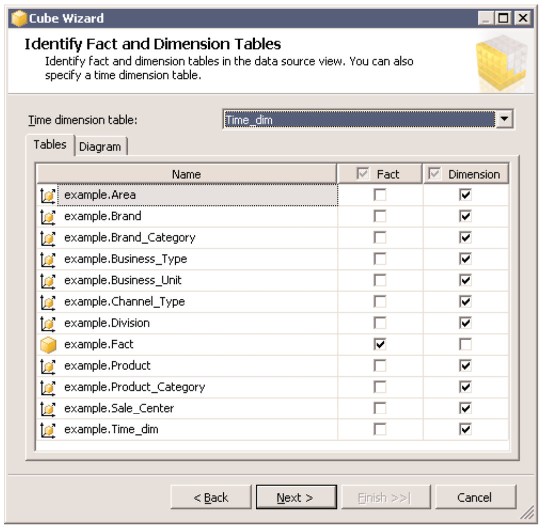
Named Calculations because these may have a significant negative effect on query performance (i.e., not handled by the define AJI in some cases). So, implementing the cast in a Teradata Database view is best where possible. Testing of the query against Teradata Database should be done when using a DSV Named Calculation.

Best Practice: DBAs and BIDS modelers should review and understand join columns and data type relationships for correctness, but more importantly, to ensure Analysis Services will generate queries that will be covered be the AJI definition.

Figure 11 represents your analytic after creating the appropriate relationships.

Defining a Cube

A wizard is available to define Analysis Services cubes based on a Data Source View. The cube wizard is an easy way to create a first prototype design but, once you become more comfortable with Analysis Services, you likely will find it more efficient to design dimensions and cubes using their specific editor.



Cube Wizard

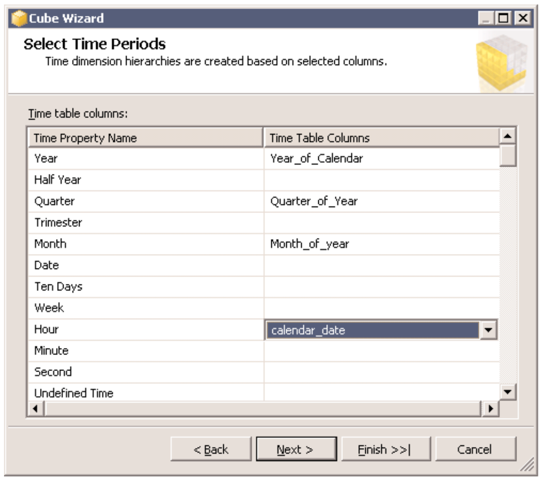
Identify Fact and Dimension Tables

Identify fact and dimension tables in the data source view. You can also specify a time dimension table.

Time dimension table: Time_dim

Name	Fact	Dimension
example.Area	☐	☑
example.Brand	☐	☑
example.Brand_Category	☐	☑
example.Business_Type	☐	☑
example.Business_Unit	☐	☑
example.Channel_Type	☐	☑
example.Division	☐	☑
example.Fact	☑	☐
example.Product	☐	☑
example.Product_Category	☐	☑
example.Sale_Center	☐	☑
example.Time_dim	☐	☑

Figure 12.



Cube Wizard

Select Time Periods

Time dimension hierarchies are created based on selected columns.

Time table columns:

Time Property Name	Time Table Columns
Year	Year_of_Calendar
Half Year	
Quarter	Quarter_of_Year
Trimester	
Month	Month_of_year
Date	
Ten Days	
Week	
Hour	calendar_date
Minute	
Second	
Undefined Time	

Figure 13.

Using the Cube Wizard

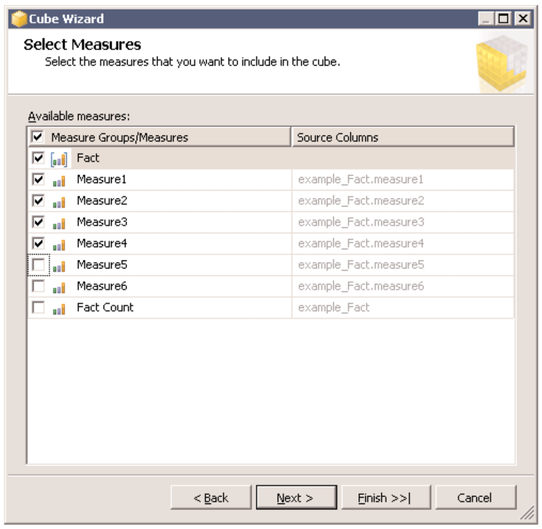
Once relationships are established, use the Cube Wizard to create a cube quickly and easily. The Cube Wizard guides you through the steps to specify the data source view and measures in the cube. When you create/design the cube, you can add existing dimensions or create new dimensions that structure the cube. You can also create dimensions separately, using the Dimension Wizard, and then add them to a cube.

Select Fact and Dimensions

Figure 12 shows the **Identify Fact and Dimension Tables** page of the wizard, with fact and dimension tables selected for the Analysis Services project. As well as selecting your Time dimension and hierarchy (see Figure 13).

Select Measures

Next, the wizard selects measures as all numeric columns in the fact table that are not linked to dimensions (see Figure 14). However, not all numeric columns may be actually measured. Though not in this example, numeric key values could link with dimension tables.



Cube Wizard

Select Measures

Select the measures that you want to include in the cube.

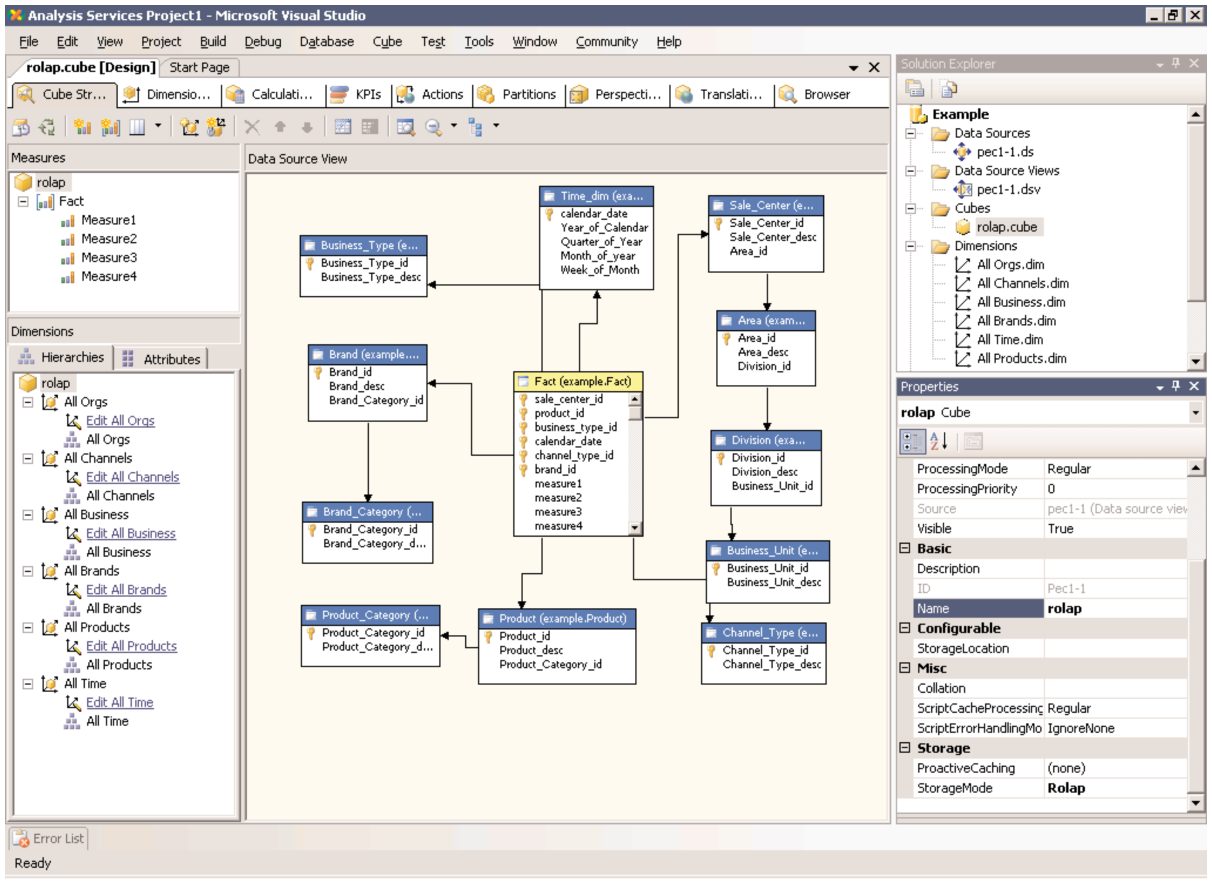
Available measures:

Measure Groups/Measures	Source Columns
☑ Fact	
☑ Measure1	example_Fact.measure1
☑ Measure2	example_Fact.measure2
☑ Measure3	example_Fact.measure3
☑ Measure4	example_Fact.measure4
☐ Measure5	example_Fact.measure5
☐ Measure6	example_Fact.measure6
☐ Fact Count	example_Fact

Figure 14.

TERADATA
Raising Intelligence

Improve Your OLAP Environment with Microsoft and Teradata



ProcessingMode	Regular
ProcessingPriority	0
Source	pec1-1 (Data source view
Visible	True
Basic	
Description	
ID	Pec1-1
Name	rolap
Configurable	
StorageLocation	
Misc	
Collation	
ScriptCacheProcessing	Regular
ScriptErrorHandlingMo	IgnoreNone
Storage	
ProactiveCaching	(none)
StorageMode	Rolap

Figure 15.

Detecting Hierarchies

Next, if you selected the **Auto build** option earlier in the wizard, the wizard can scan for hierarchies and create the hierarchy and dimension automatically. Or you can come back later to create them using the Solution Explorer Dimension Wizard.

Completing the Wizard

In Solution Explorer, in the Analysis Services project, the Analysis Services cube/model appears in the **Cubes** folder, and dimensions appear in the **Dimensions** folder. Also, in the center of the development environment, Cube Designer displays our Analysis Services cube.

Figure 15 shows the dimensions and fact tables in the designer. Notice that the fact table is yellow and the dimension tables are blue.

Reviewing Cube and Dimension Properties

After you use the Cube Wizard to define a cube, you can review your model design results in Cube Designer. You can review the structure of your cube in the Analysis Services project to understand the properties of the dimensions and the cube as defined by the Cube Wizard. Using the Cube Designer Tabs, you can view and edit various properties of a cube.

TERADATA
Raising Intelligence

Designing Partition Storage and Aggregations

After you save your Analysis Services project, you can design the storage and aggregation mode for the analytic on the **Partitions** tab in Cube Designer. For our ROLAP solution, the following should be set:

> Cube storage options
- Set Storage Mode to ROLAP for the fact/transaction table.
- Disable Pro-active caching.
- Do not enable ROLAP aggregations (AJI will supply performance).

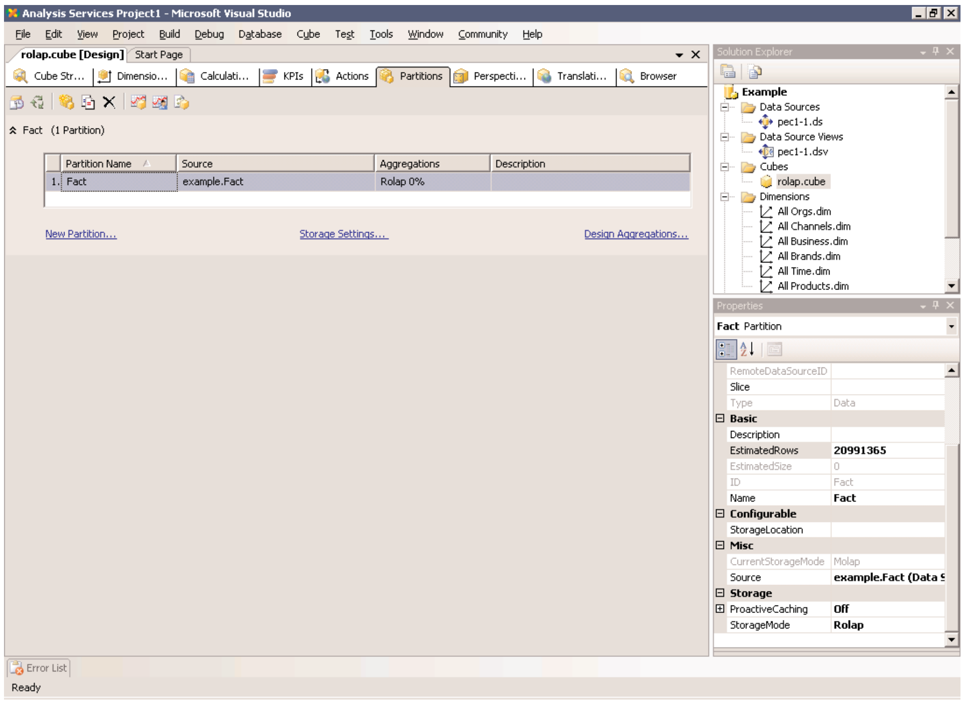
> Single partition specified for all data. Note that each Measure Group has a partition. Think of a partition as having a 1:1 relationship with a relational fact table.

Figure 16 is from **Partition tab**, where Storage Settings and Design Aggregations activities are performed.

Notice that the cube cannot be browsed because it has not been deployed yet and processed to an instance of Analysis Services.

At this point, the cube in the Analysis Services project is just a definition of a cube, which you can deploy to any instance of Analysis Services. When you deploy and process a cube, you create the defined objects in an instance of Analysis Services and populate the objects with data from the underlying data sources. In our case, using ROLAP partition storage, data will be queried only for dimensional members. Cube processing involves only a metadata reference to the underlying fact table(s).

Best Practice: Store dimensional data as MOLAP in Analysis Services. This could be different at your site depending on customer requirements or refresh rates on dimensional data. Generally, the data are small and query times are negligible.


Analysis Services Project1 - Microsoft Visual Studio
File Edit View Project Build Debug Database Cube Test Tools Window Community Help
rolap.cube [Design] Start Page
Cube Str... Dimensio... Calculati... KPIs Actions Partitions Perspecti... Translati... Browser
Solution Explorer
Example
Data Sources
pec1-1.ds
Data Source Views
pec1-1.dsv
Cubes
rolap.cube
Dimensions
All Orgs.dim
All Channels.dim
All Business.dim
All Brands.dim
All Time.dim
All Products.dim
New Partition...
Storage Settings...
Design Aggregations...
Error List
Ready

Fact (1 Partition)

	Partition Name	Source	Aggregations	Description
1.	Fact	example.Fact	Rolap 0%	

Properties: Fact Partition

RemoteDataSourceID	
Slice	
Type	Data
Basic	
Description	
EstimatedRows	20991365
EstimatedSize	0
ID	Fact
Name	Fact
Configurable	
StorageLocation	
Misc	
CurrentStorageMode	Molap
Source	example.Fact (Data S
Storage	
ProactiveCaching	Off
StorageMode	Rolap

Figure 16.

Raising Intelligence

Browsing Cube Data

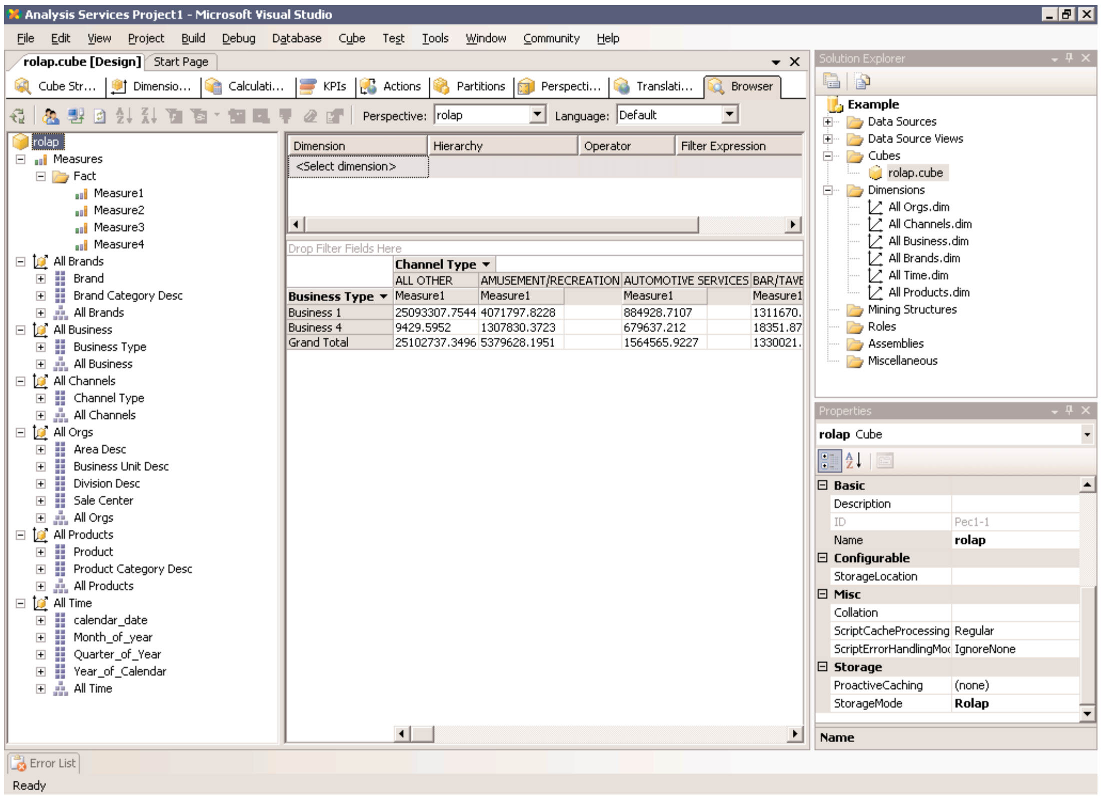
Before you browse a cube, you must process it. After you process it, open the **Browser** tab of Cube Designer to browse ROLAP cube data. The **Browser** tab has three panes – the Metadata pane, the Filter pane, and the Data pane. Use the Metadata pane to examine the structure of the cube in tree format. Use the Filter pane at the top of the **Browser** tab to define any subcube you want to browse. Use the Data pane to examine the data and drill down through dimension hierarchies (see Figure 17).

Best Practice: Set default members to define/limit the scope of initial queries to improve performance. For example, the default member in the Year attribute can be defined as the most recent year, thereby limiting queries to that year (unless otherwise specified).

SQL Profiler

SQL Profiler is a graphical application that allows system administrators to monitor Analysis Services. SQL Profiler is one of the client applications installed with SQL Server 2005 and is an invaluable tool to diagnose and troubleshoot long running queries. It reveals precisely what MDX is sent from the client application (i.e., Microsoft Excel®) and how this is translated to the SQL sent to Teradata.

Best Practice: Use Microsoft and/or Teradata tools to capture the SQL being sent by Analysis Services, and check via EXPLAIN command if query plan references the defined AJI.



	Channel Type			
	ALL OTHER	AMUSEMENT/RECREATION	AUTOMOTIVE SERVICES	BAR/TAVE
Business Type	Measure1	Measure1	Measure1	Measure1
Business 1	25093307.7544	4071797.8228	884928.7107	1311670.
Business 4	9429.5952	1307830.3723	679637.212	18351.87
Grand Total	25102737.3496	5379628.1951	1564565.9227	1330021.

Basic	
Description	
ID	Pec1-1
Name	**rolap**
Configurable	
StorageLocation	
Misc	
Collation	
ScriptCacheProcessing	Regular
ScriptErrorHandlingMo	IgnoreNone
Storage	
ProactiveCaching	(none)
StorageMode	**Rolap**

Name

Figure 17.

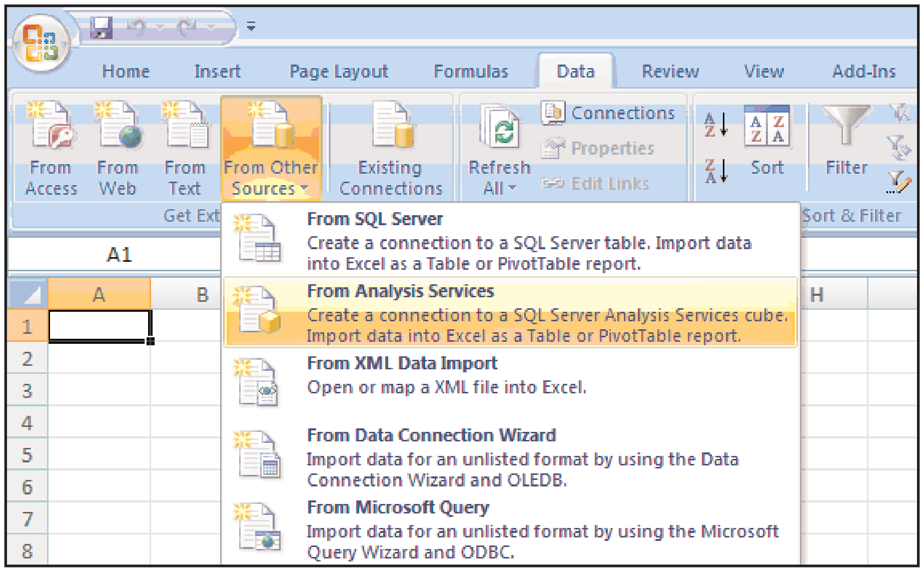
Raising Intelligence

Improve Your OLAP Environment with Microsoft and Teradata

Microsoft Office Excel 2007 Usage

Microsoft Office Excel 2007 provides full support for Microsoft SQL Server™ 2005 Analysis Services with analysis and visualization tools to help you analyze information and spot trends. Once the cube is deployed, it can be browsed in Excel just as any other cube. To create the initial connection, select the **From Other Sources** item from the data tab as illustrated in Figure 18.

The information entered here can be saved in a connection file and distributed so that regular users need not be familiar with the technical details. Once connected, users can create their own spreadsheets and charts (see Figure 19).



From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.

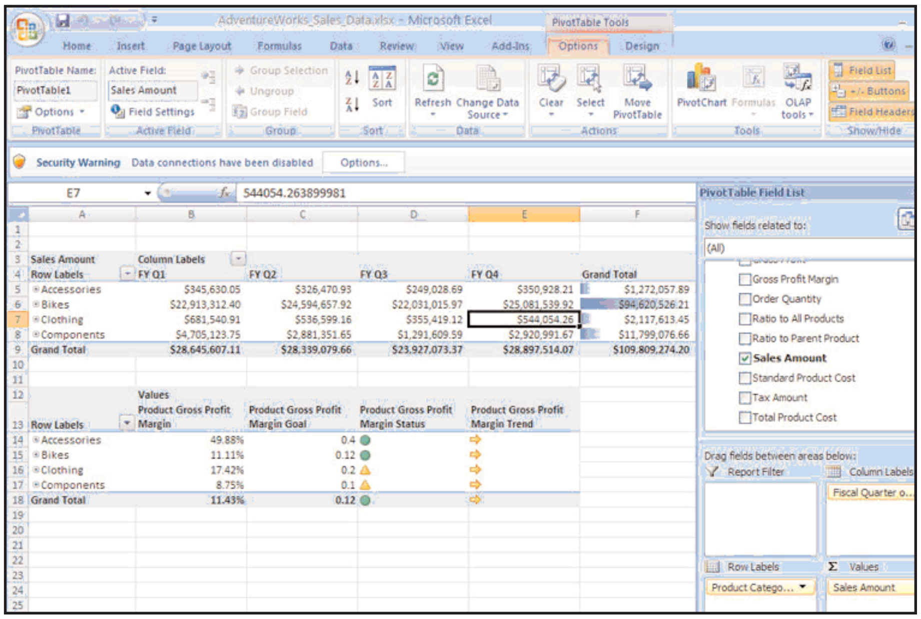
From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report.

From XML Data Import
Open or map a XML file into Excel.

From Data Connection Wizard
Import data for an unlisted format by using the Data Connection Wizard and OLEDB.

From Microsoft Query
Import data for an unlisted format by using the Microsoft Query Wizard and ODBC.

Figure 18.



Security Warning Data connections have been disabled

E7 544054.263899981

Sales Amount	Column Labels				
Row Labels	FY Q1	FY Q2	FY Q3	FY Q4	Grand Total
Accessories	$345,630.05	$326,470.93	$249,028.69	$350,928.21	$1,272,057.89
Bikes	$22,913,312.40	$24,594,657.92	$22,031,015.97	$25,081,539.92	$94,620,526.21
Clothing	$681,540.91	$536,599.16	$355,419.12	$544,054.26	$2,117,613.45
Components	$4,705,123.75	$2,881,351.65	$1,291,609.59	$2,920,991.67	$11,799,076.66
Grand Total	$28,645,607.11	$28,339,079.66	$23,927,073.37	$28,897,514.07	$109,809,274.20

Row Labels	Values Product Gross Profit Margin	Product Gross Profit Margin Goal	Product Gross Profit Margin Status	Product Gross Profit Margin Trend
Accessories	49.88%	0.4	●	⇨
Bikes	11.11%	0.12	●	⇨
Clothing	17.42%	0.2	⚠	⇨
Components	8.75%	0.1	⚠	⇨
Grand Total	11.43%	0.12	●	⇨

Gross Profit Margin
Order Quantity
Ratio to All Products
Ratio to Parent Product
[✓ Sales Amount]
Standard Product Cost
Tax Amount
Total Product Cost

Figure 19.

TERADATA
Raising Intelligence

Better Relational OLAP Response Using AJI

Here is a typical Microsoft Analysis Services ROLAP query that the Teradata Optimizer will rewrite to use the AJI. Hence, there is no additional work required for relational OLAP queries from taking advantage of the created AJI. This will occur automatically via the Teradata Optimizer without any intervention from the end user, or BI or DBA administrator. Response to this query now takes sub-seconds versus minutes in our example.

```
SELECT
    SUM ("example_Fact"."measure1") AS "example_Fact
    measure10_0",
    "example_Business_Type_2"."Business_Type_desc" AS
    "example_Business_Type1_0",
    "example_Channel_Type_3"."Channel_Type_desc" AS
    "example_Channel_Type2_0",
    "example_Product_5"."Product_Category_id" AS
    "example_Product11_1"
FROM
    "example"."Fact" AS "example_Fact",
    "example"."Business_Type" AS
    "example_Business_Type_2",
    "example"."Channel_Type" AS
    "example_Channel_Type_3",
    "example"."Product" AS "example_Product_5"
WHERE
    (("example_Fact"."business_type_id"="example_
    Business_Type_2"."Business_Type_id")
    AND
    ("example_Fact"."channel_type_id" = "example_
    Channel_Type_3"."Channel_Type_id")
    AND
    ("example_Fact"."product_id" =
    "example_Product_5"."Product_id"))
GROUP BY
    "example_Business_Type_2"."Business_Type_desc",
    "example_Channel_Type_3"."Channel_Type_desc",
    "example_Product_5"."Product_Category_id"
```

Explain below shows how the AJI would be referenced in the query plan.

Explanation

1) First, we lock a distinct EXAMPLE."pseudo table" for read on a RowHash to prevent global deadlock for **EXAMPLE.AJI_Example**.

2) Next, we lock a distinct example."pseudo table" for read on a RowHash to prevent global deadlock for example.example_Channel_Type_3.

3) We lock a distinct example."pseudo table" for read on a RowHash to prevent global deadlock for example.example_Business_Type_2.

4) We lock EXAMPLE.AJI_Example for read, we lock example.example_Channel_Type_3 for read, and we lock example.example_Business_Type_2 for read.

5) We do an all-AMPs RETRIEVE step from example.example_Business_Type_2 by way of an all-rows scan with no residual conditions into Spool 4 (all_amps), which is duplicated on all AMPs. The size of Spool 4 is estimated with high confidence to be 120 rows. The estimated time for this step is 0.03 seconds.

.

.

.

Summary

We hope this paper has shown you an approach to use Microsoft SQL Server Analysis Services and the Teradata Database Aggregate Join Index feature for a ROLAP Solution. In our business case example, we created one broad AJI to address some of the challenges in delivering deeper and wider analytics. Many combinations and various AJI constructs can greatly improve your OLAP experience. This business case is only one example. Understanding the appropriate 'cut-point' (depth and breadth of your AJI) will be key to implementing and deploying the appropriate ROLAP environment for your end users. Some of the highlighted conclusions and benefits are:

Conclusions

> AJIs can greatly improve query performance – for a relational request sent by Microsoft Analysis Services OLAP application.

> Easy to define – fairly straight forward to create AJI syntax and SQL statement.

> Network traffic reduced – since ROLAP keeps all data in the data warehouse, fewer data are transferred.

> Cube (analytical) build times minimized – eliminates data transfers, means faster delivery of analytics to consumers.

> Built in parallel – using Teradata for heavy lifting when building AJIs.

> Requires Teradata Database V2R5.1 or greater.

Benefits of Using AJIs

> Indexes created can be relatively small structures – dependent upon number of demographics rather than number of rows in Fact/base table(s).

> AJIs can be shared by multiple cube definitions – transparent to any tool or user.

> Analytics can be wider and deeper – more dimensions and more data.

Appendix A – Implementing ROLAP Cube off of Normalized Model

Many of the steps will remain the same for building this ROLAP solution from a normalized model. Although, building the AJIs off of your normalized tables will affect your load processes due to issues with AJIs. Teradata Multiload not supporting aggregate join indexes will most likely be the biggest issue as most customers load data on a nightly basis using Multiload. Here are the different load scenarios to consider:

> Multiload into normalized, and then build AJIs off of the normalized tables. This will add time to your overall load processes. This may be alright as long as the mload plus the aggregate build fits within your batch window. We highly recommend that statistics are collected on the JOIN columns and PRIMARY INDEXES of the aggregate join index DDL to get an efficient EXPLAIN plan. This will help to minimize the amount of time it takes to create the aggregates. Initial implementations, of aggregate builds on production data, have resulted in AJIs that build in less than one hour.

> Teradata Tpump into normalized tables. This will avoid the dropping and recreating of the AJIs. As updates are executed against the base table, the Teradata Database will automatically update any AJI that has the base table defined within its CREATE JOIN INDEX DDL. A thorough performance analysis will need to be executed to determine if the extra updates to the AJI can be supported in a timely manner by Tpump.

> Teradata Fastload into staging tables, and then insert select into base tables. This should be the most optimal solution as far as updating all tables and indexes involved, but this will require a rewrite of existing load processes.

Given the updating of data can be supported for the aggregate data, the presentation of data will need to be determined. Most OLAP tools work well with star schemas. Creating a star representation of data using views off of the normalized tables can be done easily. The same SQL that is used for the AJI DDL can be used for the SELECT portion of the CREATE VIEW DDL. Or tools that support normalized data model queries can simply work off of the normalized tables.

Appendix B – General Teradata Considerations

This section covers common database administration practices and considerations. Readers are encouraged to view this section as steps or tasks when investigating improvements not only in their hybrid/relational OLAP reporting environment, but also general ad hoc/managed reporting environment. They should also refer to the Teradata Database documentation for more information.

Aggregate Join Index

Question: If I have an AJI (covering store/item/day level) built against a base table(s), and I select a summarization that's at a higher level of aggregation than the AJI (i.e., division/item/day), will the optimizer use the AJI (assume the division is incorporated into the AJI)?

Taking that a step further, if I'm getting a large volume of queries summarizing at some higher levels (i.e., division/brand/week), which indicate the need for an additional AJI at a higher level, will the process to build the higher level AJI utilize the lower level AJI that's in place, or will it need to go to the base table to build?

Answer: Yes, provided division is included in AJI definition if dimension is star schema. Otherwise, for snowflake dimension schema just store, provided there is RI between dimension tables for each level.

As for the second part, yes, higher level AJI definitions will utilize lower level AJI definitions. However, prior to Teradata Database V2R6.1, the optimizer will always select from the first AJI that can satisfy the query based on build order. Therefore, you need to build the higher level first then the lower level second. As of V2R6.1, all indexes are considered, and the one with the best plan (generally the smallest one) is used.

Multi-value Compression

Multi-value compression was implemented on the Fact Table except for those columns that were defined as primary keys or foreign keys in the referential integrity constraints.

Primary Index Considerations for Data Distribution

Teradata Database is a parallel database system in which the data are distributed across a set of nodes, called AMPs. Teradata Database processes queries in parallel, with each AMP doing a portion of the work.

A primary index is required for all Teradata Database tables. If you don't assign a primary index explicitly when you create a table, then the system assigns one automatically. Data accessed using a primary index is always a single-AMP operation because a row and its primary index are stored together in the same structure. This is true whether the primary index is unique or nonunique, and whether it is partitioned or nonpartitioned.

The primary index has four purposes:

1) To define the distribution of the rows to the AMPs.

The Teradata Database distributes table rows across the AMPs based upon the hash of their primary index value. The determination of which hash bucket, and hence which AMP the row is to be stored on, is made based solely on the value of the primary index.

The choice of columns for the primary index affects how even this distribution is. An even distribution of rows to the AMPs is usually of critical importance in picking a primary index column set.

2) To provide access to rows more efficiently than with a full table scan.

If the values for all the primary index columns are specified in a DML statement, single-AMP access can be made to the rows using that primary index value.

With a partitioned primary index, faster access is also possible when all the values of the partitioning columns are specified, or if there is a constraint on partitioning columns.

Other retrievals might use a secondary index, a hash or join index, a full table scan, or a mix of several different index types.

3) To provide for efficient joins.

If there is an equality join constraint on the primary index of a table, it may be possible to do a direct join to the table (that is, rows of the table might not have to be redistributed, spooled, and sorted prior to the join).

4) To provide a means for efficient aggregations.

If the GROUP BY key is on the primary index of a table, it is often possible to perform a more efficient aggregation.

Collecting Statistics

One of the most important sources of information the Teradata Optimizer uses for choosing access plans is the set of statistics Teradata collects about the system and the data in it. Database statistics include data demographics, such as the number of rows in each table, the number of unique values within each column, and the skew and distribution of values within each column. The optimizer uses this information when selecting the best access plan from among all the possible access plans that could satisfy a given query.

It is important for the statistics to be updated as frequently as practical. Whenever new data are loaded, new indexes are built, or any other significant change occurs to data, the statistics should be updated to reflect the change. Note that out-of-date statistics can be worse than not collecting statistics at all.

In Teradata Database, statistics are particularly important because the optimizer does not allow the user to override its decisions through constructs such as hints or rewritten SQL. The optimizer will select the best plan for a given query, but its decisions are only as good as the information it uses to make them.

Connectivity

.NET Data Provider for Teradata is the advocated interface for Microsoft Analysis Services 2005. .NET Data Provider for Teradata conforms to ADO.NET 2.0 Specification. It implements and supports all required ADO.NET 2.0 interfaces and classes. BIDS displays the .NET Data Provider for Teradata seamlessly for SQL Server Analysis Services.

Installation

> The .NET Data Provider for Teradata is dependent on Microsoft .NET Framework version 2.0, which must be installed on any computer running .NET Data Provider for Teradata.

> .NET Data Provider for Teradata dependent components:

- Shared ICU Libraries for Teradata (tdicu)
- Teradata Generic Security Services (TeraGSS)
- Teradata Call-Level Interface version 2 (CLIv2)

> CLI and ODBC share one DLL namely TeraSSO.dll. Therefore, the two copies of TeraSSO.dll (ODBC installs it in System32 directory) must be identical. ODBC and CLI must be from the same TTU.

> CLI and ICU modify the path. Therefore, services such as IIS/SSAS must be restarted to pick up the new PATH.

> After installation, you can use NQuery.exe to verify installation and connectivity.

> For more information, reference .NET Data Provider for Teradata Release Definition documentation at: www.teradata.com

> Additional information on ADO.NET: http://msdn.microsoft.com/data/ref/adonet/

Implementation

BIDS Connection Manager defines and sets the provider-specific connection information for accessing the data source. We encourage you to reference the .NET Data Provider for Teradata Help for more information about some of the settings mentioned here, available under Program File→.NET Data Provider for Teradata→Help.

> ***Server Name/Data Source***
.NET Data Provider for Teradata uses CLI to connect to the Teradata Database. CLI does resolve COP names to IP addresses. However CLI's Load Balancing, COP Discovery, and Name resolution scheme is not the same as OLE DB or ODBC. For example, CLI cannot handle tdsyCOP1 because it always

appends COP1 to the name. In this example, CLI will try to resolve tdsyCOP1COP1.

The bottom line is this: use Teradata system (TDPID or DBC-Name) name.

> ***User Name/User ID***

The user account exposed to the tool via the semantic view layer should have permissions scoped appropriately versus wide system access (i.e., dbc) to provide the best performance in displaying the objects to the BI administrator tool (i.e., modeling tool). DBAs should work closely with the BI Administrator to determine the appropriate objects required to meet analytic reporting needs.

> ***Response Buffer Size (8K to 64K)***

To maximize the .NET Data for Teradata response buffers used for SQL requests and data retrieval. Administrators are encouraged to perform their own due diligence on benefits when setting from 8K to 64K with the read ahead option. With the understanding this allocates additional resources. This will help in large answer set retrieval. The difference in performance may be considerable if the result set is large. If the Row Size of the result set is large, and the Number of Rows in the result set is also large, then better performance can be expected by increasing the Maximum Response Buffer. With small result sets, the difference in performance is negligible.

> ***Read Ahead***

True enables additional buffering of results, while the current buffer is consumed by an application. This is useful when executing queries that return large result sets.

> ***Use X Views***

True limits the schema data to objects associated with the requesting user, such as objects the user owns, is associated with, has been granted privileges on, or is assigned, a role which has privileges.

General recommendation is to set to False (i.e., NOT use Use X Views), due to overhead on security check calls, but instead ensure user name has appropriate access. Hence, an appropriate Semantic Layer is critical.

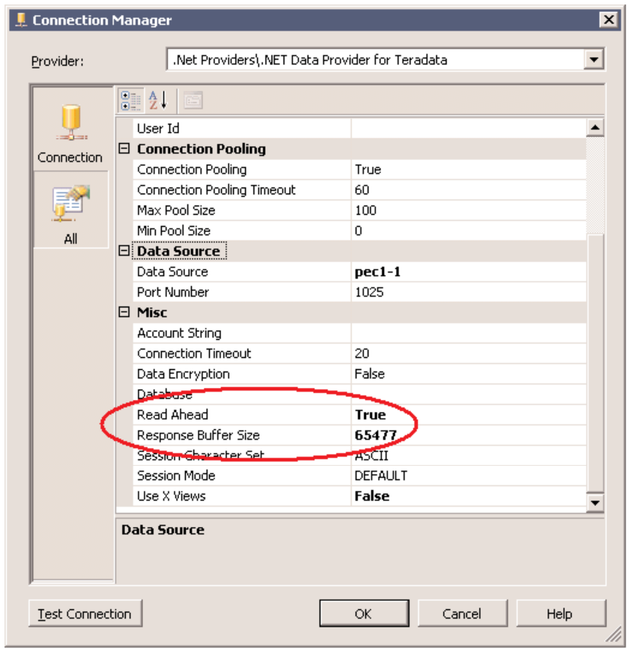
Figure B-1 shows how these settings can be set in BIDS Connection Manager.

Logging

See .NET Data Provider for Teradata Help for section called Logging and Tracing. It can be enabled at the application level.

Note: The above Connectivity tasks reflect the .NET Data Provider for Teradata. Similar tasks (e.g., Server Name and User Name) exist when using the OLE DB Provider for Teradata with the expectation of using Extended Properties dialog box for defining provider-specific settings. Please refer to the OLE DB Provider for Teradata documentation for more information.

Appendix C – General Analysis Services



Provider: .Net Providers\.NET Data Provider for Teradata

User Id	
Connection Pooling	
Connection Pooling	True
Connection Pooling Timeout	60
Max Pool Size	100
Min Pool Size	0
Data Source	
Data Source	**pec1-1**
Port Number	1025
Misc	
Account String	
Connection Timeout	20
Data Encryption	False
Database	
Read Ahead	**True**
Response Buffer Size	**65477**
Session Character Set	ASCII
Session Mode	DEFAULT
Use X Views	**False**

Data Source

Figure B-1.

Considerations

Analysis Services Known Issues

Filtering on columns declared as DATE data type does not work. Teradata will return one of these two errors:

> Error Number 5407: Invalid operation on an ANSI Datetime or Interval value

> Error Number 2666: Invalid date supplied for %TVMID.%FLDID.

SQL Server 2005 Analysis Services only supports DATETIME data types, and it does not track the Data Source Column Type (i.e., DATE vs. TIMESTAMP). It reads all DATE columns as DATETIME data values. DATE data type is converted to DATETIME by appending 12:00:00am to the date value. For example, a DATE value of “2006-12-30” is converted to “2006-12-30 12:00:00am”. At query time, SQL Server 2005 Analysis Services passes DATETIME (a.k.a. TIMESTAMP) to Teradata Database. Teradata Database cannot convert the TIMESTAMP to DATE. Teradata Database returns error 5407 or 2666 back to SSAS.

To work around this issue, either:

> Create a VIEW in Teradata Database, and cast the DATE column to TIMESTAMP(6). Modify the Analysis Services Project to use the View and not the Table.

> Use BIDS to add a Named Calculation as described in http://msdn2.microsoft.com/en-us/library/ms174859.aspx. <Expression> must be a CAST to TIMESTAMP(6); for example CAST(TxDate as TIMESTAMP(6)).

> Create/use surrogate keys, filter on a surrogate key value and displaying the actual date value (i.e. date_id, date_desc).

Best Practice: Designing a Date dimension table is a best practice for SSAS. This allows for separate columns of each date component (day of week name, month name, month number, or fiscal period) as well as other non-date attributes, such as season or corporate holiday. We strongly recommend the practice of designing a table or view in the database with a surrogate key and columns in Teradata Database.

SQL Server 2005 SP2 is Recommended (Required for 64-bit Environments)

Though the comprehensive list of fixes included in SP2 was not evaluated in this paper, it is clear that without SP2 ROLAP, performance and connection capability will be impacted. Once SP2 is made generally available, the installation of SP2 should be considered a requirement for SSAS implementations and will address these observed behaviors.

> SP1 has an issue with submitting too many queries to the database per ROLAP request. The actual request translates to one query for all base table dimensions plus one query per calculated measure defined in Analysis Server.

> Connectivity between Analysis Services 2005 and Teradata Database was implemented using OLE DB Provider for Teradata. The .NET Data Provider for Teradata did not work with this solution as it does not support parameterized requests with the ‘@’ sign representing the parameter within a query. A post-SP2 version (contact your Microsoft support representative) supports the .NET Data Provider for Teradata which requires the ‘?’ as the value representing a parameter.

TERADATA
Raising Intelligence

Improve Your OLAP Environment with Microsoft and Teradata

Optimizing SQL Query Generation through SSAS Property Setting

Observed Behavior

The WHERE clause of the SQL query generated by SSAS is impacted by the nature of the KeyColumns property of Attribute Dimension. When this property is a single column reference, the WHERE clause references only this column using the '?' parameter indicator (see Figure C-1).

This, however, is an inadequate reference when querying Teradata Database or any database for members below this level.

For example: a query involving the Week level of a time dimension will produce a query whose WHERE clause is formatted something like – (WHERE dwh_date.week_id = ?). The result is a scan of a much larger data set than necessary if the database is indexed on, for instance, Year.

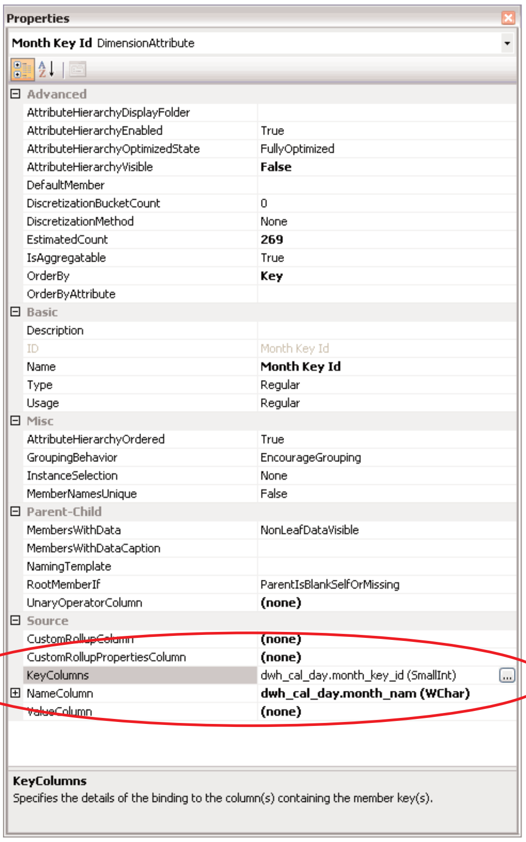
A better query would include a WHERE clause that included a Year value: (WHERE dwh_date.year_id = 106 and dwh_date.week_id = ?).

Corrective Action

In the KeyColumns property, include a Column Binding reference to the higher level columns supporting the user hierarchy of instance. Adding a reference to the Year_id column and moving this reference to the top of the collection produces a query that references each level defined in the KeyColumns collection (see Figure C-2).

Role Playing Dimensions

If the customer requires reporting against multiple dates (e.g., contract date, open date, and close date) against information such as miles driven, contract amount, and surcharge, it can all be viewed based upon a date hierarchy that can join to the transaction table based upon any of these three dates. Rather than create three copies of a date hierarchy table that would join on each individual date, role playing dimensions were defined to support this type of analysis.


Properties

Month Key Id DimensionAttribute

Advanced	
AttributeHierarchyDisplayFolder	
AttributeHierarchyEnabled	True
AttributeHierarchyOptimizedState	FullyOptimized
AttributeHierarchyVisible	**False**
DefaultMember	
DiscretizationBucketCount	0
DiscretizationMethod	None
EstimatedCount	**269**
IsAggregatable	True
OrderBy	**Key**
OrderByAttribute	
Basic	
Description	
ID	Month Key Id
Name	**Month Key Id**
Type	Regular
Usage	Regular
Misc	
AttributeHierarchyOrdered	True
GroupingBehavior	EncourageGrouping
InstanceSelection	None
MemberNamesUnique	False
Parent-Child	
MembersWithData	NonLeafDataVisible
MembersWithDataCaption	
NamingTemplate	
RootMemberIf	ParentIsBlankSelfOrMissing
UnaryOperatorColumn	**(none)**
Source	
CustomRollupColumn	**(none)**
CustomRollupPropertiesColumn	**(none)**
KeyColumns	dwh_cal_day.month_key_id (SmallInt)
NameColumn	**dwh_cal_day.month_nam (WChar)**
ValueColumn	**(none)**

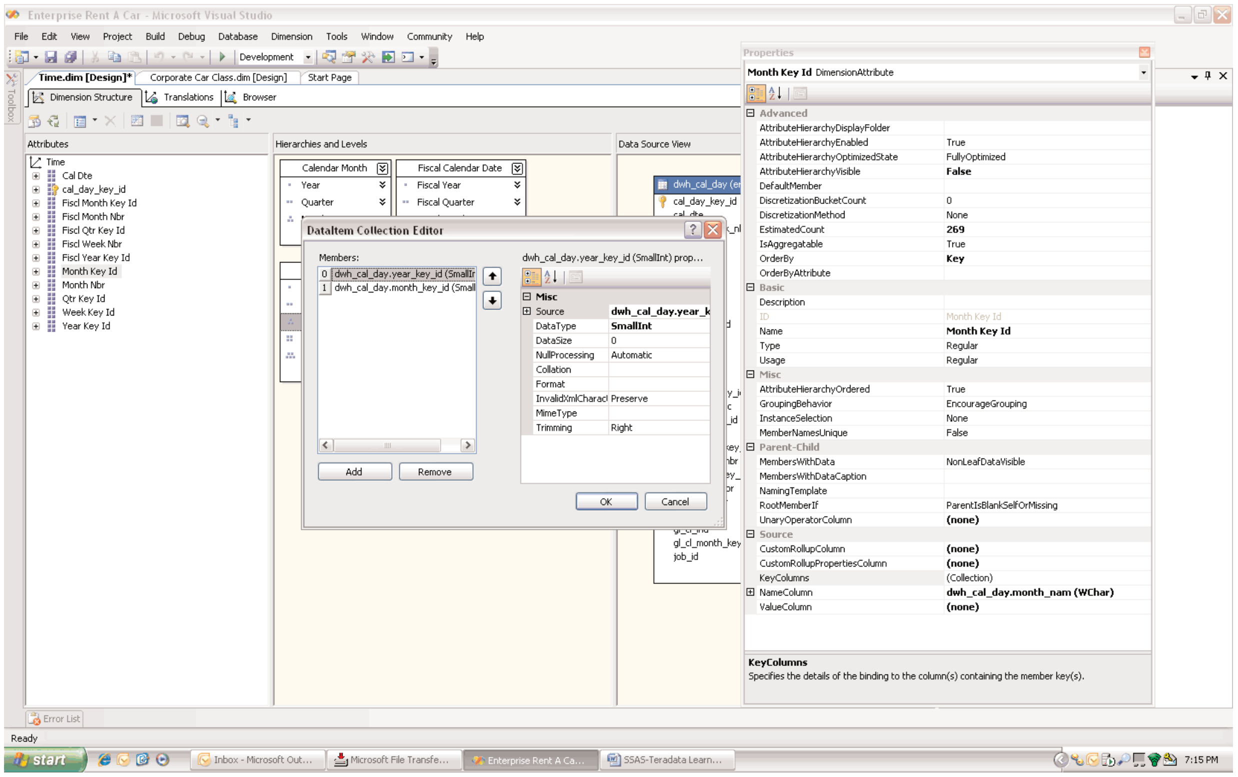
KeyColumns
Specifies the details of the binding to the column(s) containing the member key(s).

Figure C-1.

Collection References

Collection references enable the defining of relationships between a dimension table to another dimension or fact table based upon multiple columns. That is because a single column may not be enough to identify the 1:1 or 1:M relationship needed to define a valid relationship between tables. An example of this would be a transaction table that stored multiple years of historical data down to the month level. An associated dimension table might have two columns defined, a year column and a month column. If the month column only contained values of one through 12, then a

TERADATA
Raising Intelligence



Attributes: Time — Cal Dte, cal_day_key_id, Fiscl Month Key Id, Fiscl Month Nbr, Fiscl Qtr Key Id, Fiscl Week Nbr, Fiscl Year Key Id, Month Key Id, Month Nbr, Qtr Key Id, Week Key Id, Year Key Id

DataItem Collection Editor

Members:

0 dwh_cal_day.year_key_id (SmallIn
1 dwh_cal_day.month_key_id (Small

dwh_cal_day.year_key_id (SmallInt) prop...

Misc	
Source	**dwh_cal_day.year_k**
DataType	**SmallInt**
DataSize	0
NullProcessing	Automatic
Collation	
Format	
InvalidXmlCharact	Preserve
MimeType	
Trimming	Right

Add Remove OK Cancel

Properties

Month Key Id DimensionAttribute

Advanced	
AttributeHierarchyDisplayFolder	
AttributeHierarchyEnabled	True
AttributeHierarchyOptimizedState	FullyOptimized
AttributeHierarchyVisible	**False**
DefaultMember	
DiscretizationBucketCount	0
DiscretizationMethod	None
EstimatedCount	**269**
IsAggregatable	True
OrderBy	**Key**
OrderByAttribute	
Basic	
Description	
ID	Month Key Id
Name	**Month Key Id**
Type	Regular
Usage	Regular
Misc	
AttributeHierarchyOrdered	True
GroupingBehavior	EncourageGrouping
InstanceSelection	None
MemberNamesUnique	False
Parent-Child	
MembersWithData	NonLeafDataVisible
MembersWithDataCaption	
NamingTemplate	
RootMemberIf	ParentIsBlankSelfOrMissing
UnaryOperatorColumn	**(none)**
Source	
CustomRollupColumn	**(none)**
CustomRollupPropertiesColumn	**(none)**
KeyColumns	(Collection)
NameColumn	**dwh_cal_day.month_nam (WChar)**
ValueColumn	**(none)**

KeyColumns
Specifies the details of the binding to the column(s) containing the member key(s).

Figure C-2.

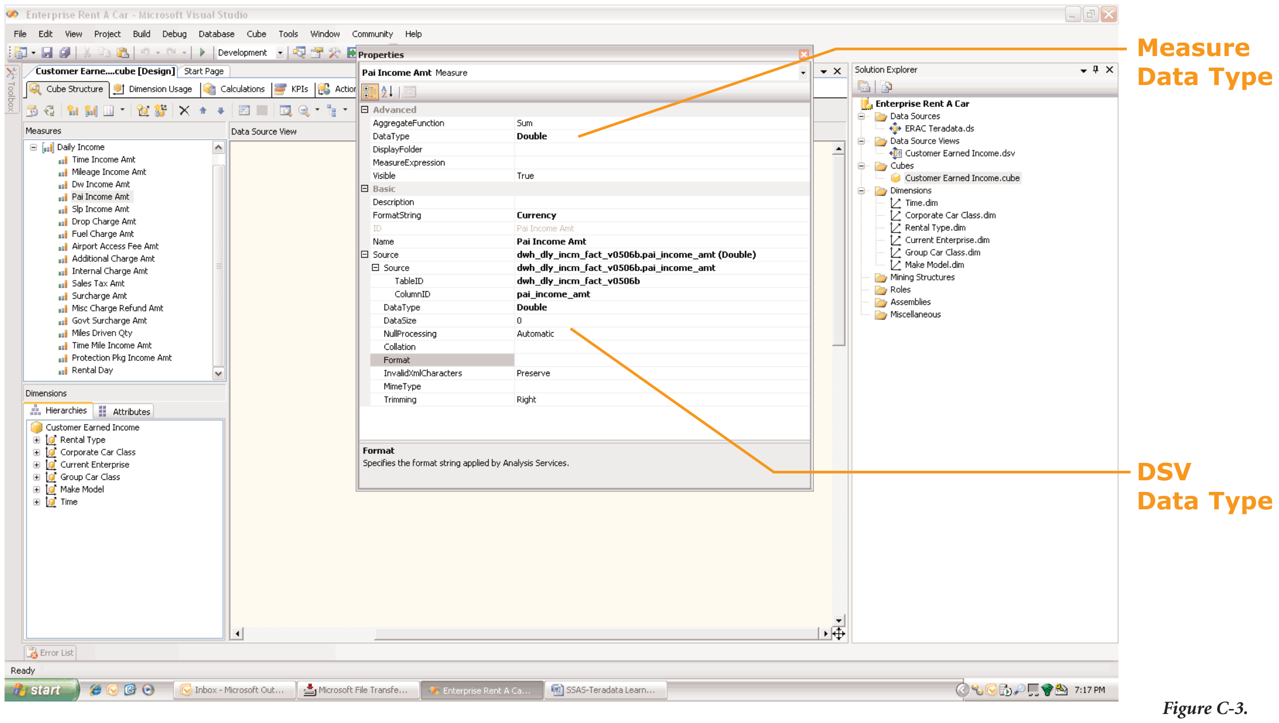
join defined only on the month column would result in an invalid M:M join based upon only the month values. In this case, it is necessary to define the join based upon two columns, year and month. This is accomplished by defining the collection references within Analysis Server. Collection references let Analysis Server know that it needs to create SQL that joins the dimension table to the transaction table using both the year and month column.

General/Miscellaneous

SQL Server Analysis Server 2005 has the capability of defining a semantic layer known as a Data Source View (DSV) within the SQL Server BIDS. It is possible to define a New Named Query within the DSV that has similar functionality as a view has within a database. We do not recommend using this feature because it greatly affects the SQL that Analysis Server generates to the database. If named queries are utilized, it may limit the ability for AJIs to be used by the optimizer to support the OLAP requests from Analysis Server.

TERADATA
Raising Intelligence

Improve Your OLAP Environment with Microsoft and Teradata



Properties

Pai Income Amt Measure

Advanced	
AggregateFunction	Sum
DataType	**Double**
DisplayFolder	
MeasureExpression	
Visible	True
Basic	
Description	
FormatString	**Currency**
ID	Pai Income Amt
Name	**Pai Income Amt**
Source	**dwh_dly_incm_fact_v0506b.pai_income_amt (Double)**
Source	**dwh_dly_incm_fact_v0506b.pai_income_amt**
TableID	**dwh_dly_incm_fact_v0506b**
ColumnID	**pai_income_amt**
DataType	**Double**
DataSize	0
NullProcessing	Automatic
Collation	
Format	
InvalidXmlCharacters	Preserve
MimeType	
Trimming	Right

Format
Specifies the format string applied by Analysis Services.

Figure C-3.

Source Column Data Type Does Not Update When DSV is Modified

Observed Behavior

Queries generated when browsing the cube returned a non-specific "query terminated with failure" message, and the query repeatedly re-executed until manually terminated.

Presumed Cause

Analysis Services Measures were originally defined based on column references to the Fact Table as defined in the DSV. As part of this activity, the Source data type property of each measure was set automatically based on the referenced table.column data type. Subsequently, the Fact Table was replaced in the Analysis Services DSV with a reference to a Teradata Database view. The data types of some columns previously defined as integers were now defined in the view as decimal(18,2). However, the Source data type property of Measures referencing these columns was not automatically updated and so now had invalid data type definitions (i.e., system.decimal values defined as integer).

Corrective Action

Each Measure's Source Data Type property was compared to the data type defined in the referenced DSV column. Where different, the property was redefined (see Figure C-3).

Raising Intelligence

Inadequate ROLAP SQL Query Formation in BIDS Analysis Services Browser (OWC), Excel 2003

Observed Behavior

SQL queries generated by the BIDS Analysis Services browser to access ROLAP do not include proper filtering for levels above the current lowest level of the MDX query.

For example, queries for values at the day level of time do not include a WHERE clause reference to year, quarter, or month.

This causes the query to process significantly more rows; creating greatly increased query times.

Corrective Action

Higher end browsers, such as ProClarity® 6.1 and SSRS, generate superior queries that do not suffer from this behavior.

Attribute Dimension Property Settings Seem to be Cached and Not Updated as Expected

Observed Behavior

SQL queries generated by the SSAS browser that include a WHERE clause parameter do not appear to properly pass the value of the defined Default Member.

Queries launched under these conditions run for unexpectedly long times and may repeatedly re-execute.

Presumed Cause

Property Changes to properties of an Attribute Dimension containing a Default Member definition are not being recognized by SSAS. So, the observed behavior recurs despite property changes made to diagnose the issue.

Corrective Action

Given that the value of the parameter is related to the KeyColumns property, the table.column reference for KeyColumns and NameColumn were modified to test the effect on the query: both were set as references to the key_id column.

With this modification, the query performed as originally expected – quickly and accurately. After completion of this test, the NameColumn property was reset to its original value (desc column), and the query continued to perform as expected.

It seems the change of both properties in a single update caused the previous (cached?) design to be updated.

Note: This may no longer be true with the final release of SQL Server 2005 SP2.

Stale Cached Data

Observed Behavior

Browsing data yields data different than recent updates in the ROLAP partition.

Corrective Action

Clear the Analysis Services data cache on a regular basis with the script:

```
<ClearCache xmlns=”http://schemas.microsoft.com/analy-
sisservices/2003/engine“>
  <Object>
    <DatabaseID>Edgars</DatabaseID>
  </Object>
</ClearCache>
```

Where Edgars is the name of the Analysis Services database containing the cube.

Raising Intelligence

Browsing MOLAP Dimensions Results in SQL Queries to the ROLAP Partitions

Observed Behavior

Browsing a dimension with no data present results in queries sent to Teradata Database even though no data are requested or shown.

The problem is that some client applications inadvertently request data when not required. If this is the case, performance can be enhanced by creating a constant calculation that does not require any data from the ROLAP source.

Sample Tests

Attach SQL Profiler to Analysis Services. Browse a dimension, and observe the MDX queries sent to Analysis Services and any generated SQL queries.

Corrective Action

Create an additional measure, say Measures.DimensionQuery, in the cube. In the cube's MDX Script, create an assignment that populates this measure with a constant value:

```
Measures.DimensionQuery = 0;
```

In the Cube Editor, set the Visible Property of this measure to False.

SQL Query Generation Produces Multiple Queries

Observed Behavior

Browsing a cube using any of the tested applications (SSAS Browser, SSRS, ProClarity) produces multiple SQL queries.

Storage is a single ROLAP partition on Fact Table. All queries executed using a slicer in Time dimension (Calendar Year 2006).

The number of queries produced seems to follow this pattern:

> One query is produced for the entire Measure Group.

> One query is produced for each Calculated Measure.

> Additional queries are produced for years other than the sliced year.

Sample Tests

Conditions:

> Replace time dimension table with view limiting time dimension to a single year (2006, levels Year/Quarter/Month/Day).

> Slice on Calendar Year 2006.

Test Results:

> w/19 stored measures in browser, one SQL query

> w/19 stored measures + one calculated member in browser, two SQL queries

> w/19 stored measures + two calculated members in browser, three SQL queries

> w/19 stored measures + 30 calculated members in browser, 31 SQL queries

Conditions:

> Replace view with dim table reference in DSV (1990 – 2012, levels Year/Quarter/Month/Day).

> Slice on Calendar Year 2006.

Test Results:

> w/19 stored measures in browser, three SQL queries

> w/19 stored measures + one calculated member in browser, five SQL queries

> w/19 stored measures + 30 calculated members in browser, many (30+) SQL queries

Corrective Action

Installed SQL Server 2005 SP2, and the behavior was eliminated without any additional change to the configuration.

TERADATA
Raising Intelligence

General Comments

> The cube browser, as well as Excel 2003, does not pass restrictions qualifying higher level hierarchy levels during expanding levels down to the leaf. In cases where the lower level members of the hierarchy have many members, the unrestricted SQL might return many more rows than expected. This can result in slow response times in the Cube Browser. This behavior is not apparent in other front-end tools, such as ProClarity or MS Reporting Services.

> Use of Calculated Measures in a Pre-SQL Service 2005 Analysis Services Service Pack 2 installation results in one query per Calculated Measure to be generated serially by Analysis Services. With many calculated measures, this can result in severely impacted performance. It is important if you are using Calculated Measures to be running with SS2K5 SP2 at a minimum.

> Analysis Services aggregate calculations may result in an overflow condition – if the resulting aggregate is based on an INTEGER data type on Teradata Database, and the result exceeds that storage capacity. Potentially, an error message is not thrown, and Analysis Services will continue to execute the query. CASTing the data type to a DEC(18) will work around this issue.

> The extensive use of parameters in the SQL generated by Analysis Services might affect the ability of the Teradata Optimizer to utilize sparse AJIs in pre-Teradata Database V2R7/V12 engines. This is not a bug. When analyzing AJI usage with Analysis Services-generated SQL, be aware that substituting literal values for AJI testing does not necessarily guarantee AJI utilization, since the optimizer will generate and cache the query plan without the knowledge of the specific value to be bound to the parameter at runtime. One work-around is to create a view with the same restrictions as in the sparse AJI. The applicability of this is situation dependent, but it's worthwhile being aware of for AJI analysis and optimization.

About the Authors

Rupal Shah

Rupal Shah is a technical consultant who has been with Teradata for 15 years. Prior to consulting on the Microsoft partnership, he spent six years as a technical consultant for the Cognos and Hyperion partnerships. Along with his extensive experience working with business intelligence partners, Rupal has worked with various Teradata application organizations for whom he provided database consulting. He received his B.A. in Math and Computer Science from the University of California at San Diego, and he is currently based in San Diego.

Richard Tkachuk

Richard Tkachuk is a Lead Program Manager on the Analysis Services team at Microsoft. His main interests are around the Analysis Services security model, MDX, and developing business intelligence applications. Richard has been with Microsoft for six years and in the Business Intelligence space for many more. Originally from Winnipeg, Canada, he now resides in Sammamish, Washington.

TERADATA
Raising Intelligence