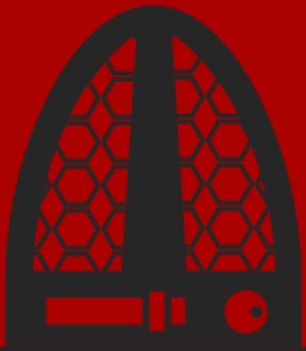


# SQL Server Analysis Services Succinctly

by Stacia Misner

Syncfusion | Technology Resource Portal

# SQL Server Analysis Services Succinctly

**By**

**Stacia Misner**

Foreword by Daniel Jebaraj

## Important licensing information. Please read.

**Technical Reviewer:** Rui Machado
**Copy Editor:** Courtney Wright
**Acquisitions Coordinator:** Marissa Keller Outten, director of business development, Syncfusion, Inc.
**Proofreader:** Graham High, content producer, Syncfusion, Inc.

# Table of Contents

# The Story behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

## Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

## Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

## The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

## The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

## Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

## Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to "enable AJAX support with one click," or "turn the moon to cheese!"

## Let us know what you think

If you have any topics of interest, thoughts, or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and "Like" us on Facebook to help us spread the word about the *Succinctly* series!

## About the Author

Stacia Misner is a Microsoft SQL Server MVP, SQL Server Analysis Services Maestro, Microsoft Certified IT Professional-BI, and Microsoft Certified Technology Specialist-BI with a Bachelor's degree in Social Sciences. As a consultant, educator, author, and mentor, her career spans more than 25 years, with a focus on improving business practices through technology. Since 2000, Stacia has been providing consulting and education services for Microsoft's business intelligence technologies, and in 2006 she founded Data Inspirations. During these years, she has authored or co-authored multiple books and articles as well as delivered classes and presentations around the world covering different components of the Microsoft SQL Server database and BI platform.

# Chapter 1  Introduction to SQL Server Analysis Services

SQL Server Analysis Services is one of several components available as part of Microsoft SQL Server 2012 that you can use to develop a business intelligence analytic solution. In this introduction to SQL Server Analysis Services, I explain the concept of business intelligence and all available options for architecting a business intelligence solution. I also review the process of developing an Analysis Services database at a high level and introduce the tools you use to build, manage, and query Analysis Services.

## What Is Business Intelligence?

Business intelligence means different things to different people. Regardless of how broadly or narrowly the term is used, a globally accepted concept is that it supports the decision-making process in organizations. In short, people at all levels of an organization must gather information about the events occurring in their business before making a decision that can help that business make or save money.

A common problem in many businesses is the inability of the operational systems gathering details about business events to facilitate the information-gathering process and consequently the decision-making process is impeded. When the only source of information is an operational system, at worst people rely on gut instinct and make ill-informed decisions because they cannot get the information they need, while at best people have tools or other people to help them compile the needed data, but that process takes time and is tedious.

Most business applications store data in relational systems, which anyone can query if they have the right tools, skills, and security clearance. Why then is it necessary to move the data into a completely different type of database? To understand this requirement and why Analysis Services is included in a business intelligence solution, it's helpful to compare the behavior of a relational engine like SQL Server with an Online Analytical Processing (OLAP) engine like Analysis Services. First, let's consider the three types of questions that are important to decision makers as they analyze data to understand what's happening in the business:

- **Summarization**. Users commonly want to summarize information for a particular range of time, such as total sales across a specified number of years.

- **Comparison**. Users want to answer questions that require comparative data for multiple groups of information or time periods. For example, they might want to see total sales by product category. They might want to break down this data further to understand total sales by product category or by all months in the current year.

- **Consolidation**. Users often also have questions that require combining data from multiple sources. For example, they might want to compare total sales with the forecasted sales. Typically, these types of data are managed in separate applications.

***Note: For the purposes of this book, I use summary, comparison, and consolidation questions to represent the business requirements for a business intelligence solution to build. Although the scenario I discuss is extremely simple, the same principles that I describe here also apply to real-world scenarios for which decision-makers have many more questions that require answers that the data can answer, if only it were structured in a better way.***

Each of these types of queries can be problematic when the data is available in a relational engine only for the following reasons:

- Queries for decision-making rely on data stored in the same database that is being used to keep the business running. If many users are executing queries that require the summarization of millions of rows of data, a resource contention problem can arise. A summarized query requires lots of database resources and interferes with the normal insert and update operations that are occurring at the same time as the business operations.

- Data sources are often focused on the present state. Historical data is archived after a specified period of time. Even if it is not archived completely, it might be kept at a summarized level only.

- Calculations often cannot be stored in the relational database because the base values must be aggregated before calculations are performed. For example, a percent margin calculation requires the sum of sales and the sum of costs to be calculated first, total costs to be subtracted from total sales next, and finally the result to be derived by dividing it by the total sales. Whether the logic is relatively simple, as with a percent margin calculation, or complex as with a weighted allocation for forecasting, that logic is not stored in the relational engine and must be applied at query time. In that case, there is no guarantee that separate users using different tools to gather data will construct the calculation in identical ways.

- Relational storage of data often uses a structure called third normal form, which spreads related data across multiple tables. As a result, the retrieval of data from these tables requires complex queries that can be difficult to write and can contain many joins that might cause queries to run slowly.

An OLAP engine solves these problems in the following ways:

- The use of a separate data source for querying reduces resource contention. Of course, you can maintain a replica of a relational database that you dedicate to reporting, but there are other reasons to prefer OLAP over relational.

- You can retain historical data in an OLAP database that might otherwise be eliminated by overwrites or archiving the source system. Again, you can resolve this problem by creating a relational data mart or data warehouse, but there are still other reasons to implement OLAP.

- A more significant benefit of OLAP is the centralization of business logic to ensure all users get the same answer to a particular query regardless of when the query is run or the tool used to execute the query.

- The storage mechanism used by Analysis Services is designed for fast retrieval of data. If you prefer to write a query rather than use a query builder tool, many times the queries are shorter and simpler (once you learn the query syntax for Analysis Services, MDX).

- OLAP databases store data in binary format, resulting in smaller files and faster access to the data.

- Last but not least, OLAP databases provide users with self-service access to data. For example, a Microsoft Excel user can easily connect to an Analysis Services cube and browse its data by using pivot charts or pivot tables.

## Architecture Options

There are several different ways that you can architect Analysis Services:

- **Prototype**. This is the simplest architecture to implement. In this case, you install Analysis Services on a server, and then create and process a database to load it with data. Your focus is on a single data load to use in a proof of concept and therefore you do not implement any data refresh processes as part of the architecture.

- **Personal or team use**. If you have a single data source with a relatively simple structure and small volumes of data, and if you have no need to manage historical changes of the data (also known as slowly changing dimensions), you can implement Analysis Services and add a mechanism for refreshing your Analysis Services database on a periodic basis, such as nightly or weekly.

- **Department or enterprise use**. As the number of users requiring access to the database grows, or the number of data sources or complexity of the data structure increases, you need to set up a more formal architecture. Typically, this requires you to set up a dedicated relational source for Analysis Services, such as a subject-specific data mart or a data warehouse that houses multiple data marts or consolidates data from multiple sources. In this scenario, you implement more complex extract, transform, and load (ETL) processes to keep the data mart or data warehouse up-to-date and also to keep the Analysis Services database up-to-date. If you need to scale out the solution, you can partition the Analysis Services database.

**Multidimensional or Tabular?**

When you install Analysis Services, you have the option to install one of the three types of instances:

- Multidimensional
- Tabular
- PowerPivot for SharePoint

The first two instance types are standalone server instances, while the third requires integration with SharePoint.

The multidimensional server hosts databases containing one or more cubes. It is a mature, feature-rich product that supports complex data structures and scales to handle high data volumes and large numbers of concurrent users.

The tabular server supports a broader variety of data sources for models stored in its databases, but manages data storage and memory much differently. For more information, see http://msdn.microsoft.com/en-us/library/hh994774.aspx.

***Note: Although the focus of this book is the multidimensional server mode for Analysis Services, the architecture for an environment that includes Analysis Services in tabular server mode is similar. Whereas a multidimensional database requires relational data sources, a tabular database can also use spreadsheets, text data, and other sources. You can use the same client tools to query the databases.***

In the **prototype** architecture, your complete environment can exist on a single server, although you are not required to set it up this way. It includes a relational data source, an Analysis Services instance, and a client tool for browsing the Analysis Services database, as shown in Figure 1. The relational data source can be a SQL Server, DB2, Oracle, or any database that you can access with an OLE DB driver. For prototyping purposes, you can use the Developer Edition of Analysis Services, but if you think the prototype will evolve into a permanent solution, you can use the Standard, Business Intelligence, or Enterprise Edition, depending on the features you want to implement as described in Table 1. For browsing the prototype database, Excel 2007 or higher is usually sufficient.

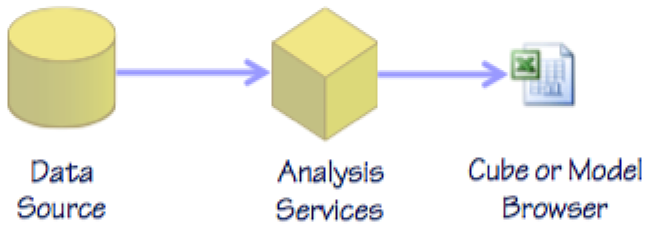


*Figure 1: Prototype Architecture*

*Table 1: Feature Comparison by Edition*

| Feature | Standard Edition | Business Intelligence Edition | Developer and Enterprise Editions |
|---|---|---|---|
| Account Intelligence | Yes | Yes | Yes |
| Actions | Yes | Yes | Yes |
| Advanced Dimensions (Reference, Many-to-Many) | Yes | Yes | Yes |
| Advanced Hierarchy Types (Parent-Child, Ragged) | Yes | Yes | Yes |
| Aggregations | Yes | Yes | Yes |
| Binary and Compressed XML Transport | Yes | Yes | Yes |
| Custom Assemblies | Yes | Yes | Yes |

| **Feature** | **Standard Edition** | **Business Intelligence Edition** | **Developer and Enterprise Editions** |
|---|---|---|---|
| Custom Rollups | Yes | Yes | Yes |
| Dimension and Cell Level Security | Yes | Yes | Yes |
| Direct Writeback | No | Yes | Yes |
| Drillthrough | Yes | Yes | Yes |
| Hierarchies | Yes | Yes | Yes |
| High Availability | Yes | Yes | Yes |
| KPIs | Yes | Yes | Yes |
| Linked Measures and Dimensions | No | Yes | Yes |
| MDX Queries and Scripts | Yes | Yes | Yes |
| Measure Expressions | No | Yes | Yes |
| MOLAP, ROLAP, and HOLAP Storage Modes | Yes | Yes | Yes |
| Multiple Partitions | Up to 3 | Yes | Yes |
| Perspectives | No | Yes | Yes |
| Proactive Caching | No | Yes | Yes |
| Programmability (AMO, AMOMD.NET, OLEDB, XML/A, ASSL) | Yes | Yes | Yes |
| Push-Mode Processing | No | Yes | Yes |
| Role-Based Security Model | Yes | Yes | Yes |
| Scalable Shared Databases (Attach/Detach, Read Only) | No | Yes | Yes |
| Scalable String Storage | Yes | Yes | Yes |
| Semi-additive Measures | LastChild only | Yes | Yes |
| Time Intelligence | Yes | Yes | Yes |
| Translations | Yes | Yes | Yes |
| Writeback Cells | Yes | Yes | Yes |

| Feature | Standard Edition | Business Intelligence Edition | Developer and Enterprise Editions |
|---|---|---|---|
| Writeback Cube | Yes | Yes | Yes |
| Writeback Dimensions | No | Yes | Yes |

For a **personal or team** solution, you introduce automation to keep data current in Analysis Services. You use the same components described in the prototype architecture: a data source, Analysis Services, and a browsing tool. However, as shown in Figure 2, you add Integration Services as an additional component to the environment. Integration Services uses units called packages to describe tasks to execute. You can then use a scheduled process to execute one or more packages that update the data in the Analysis Services database. Excel is still a popular choice as a browsing tool, but you might also set up Reporting Services to provide access to standard reports that use Analysis Services as a data source.

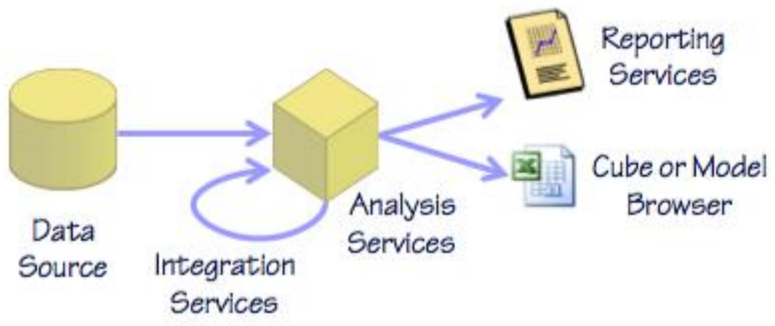


*Figure 2:  Personal or Team Architecture*

To set up an architecture for organizational use, as shown in Figure 3, you introduce a data mart or data warehouse to use as a source for the data that is loaded into Analysis Services. Integration Services updates the data in the data mart on a periodic basis and then loads data into Analysis Services from the data mart. In addition to Excel or Reporting Services as client tools, you can also use SharePoint business intelligence features, which include Excel Services, SharePoint status indicators and dashboards, or PerformancePoint scorecards and dashboards.

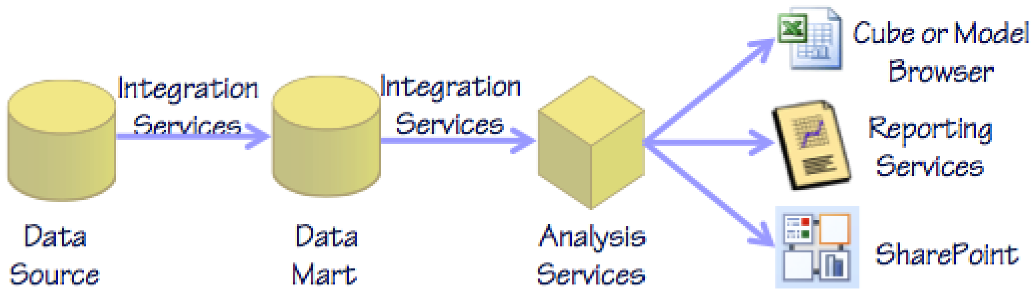


*Figure 3:  Organizational Architecture*

## Development, Management, and Client Tools

If you are responsible for creating or maintaining an Analysis Services database, you use the following tools:

- SQL Server Data Tools (SSDT)
- SQL Server Management Studio (SSMS)
- A variety of client tools

**SSDT** is the environment you use to develop an Analysis Services database. Using this tool, you work with a solution that contains one or more projects, just like you would when developing applications in Visual Studio.

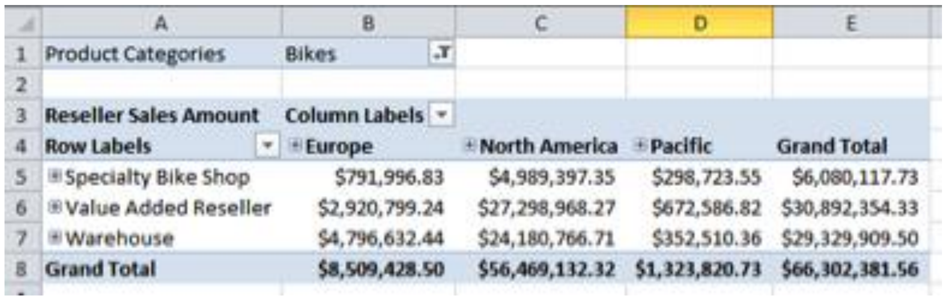
You can use **SSMS** to configure server properties that determine how the server uses system resources. You can also use Object Explorer to see the databases deployed to the server and explore the objects contained within each database. Not only can you view an object's properties, but in some cases you can also make changes to those properties. Furthermore, you can create scripts of an object's definition to reproduce it in another database or on another server.

SSMS also gives you a way to quickly check data either in the cube itself or in individual dimensions. You can use the MDX query window to write and execute queries that retrieve data from the cube. A graphical interface is also available for browsing these objects without the need to write a query.

Another feature in SSMS is the XML for Analysis (XMLA) query window in which you write and execute scripts. You can use XMLA scripts to create, alter, or drop database objects and also to process objects, which is the way that data is loaded into an Analysis Services database. You can then put these scripts into Integration Services packages to automate their execution or you can put them into SQL Server Agent jobs. However, you are not required to use scripts for processing. You can instead manually process objects in SSMS whenever necessary or create Integration Services packages to automate processing.

As part of the development process, you should use the **client tools** that your user community is likely to use to ensure the browsing experience works as intended. In this chapter, I explain the choice of client tools available from Microsoft, but there are also several third-party options to consider, and of course you can always create a custom application if you want users to have specific functionality available. The Microsoft business intelligence stack includes the following tools:

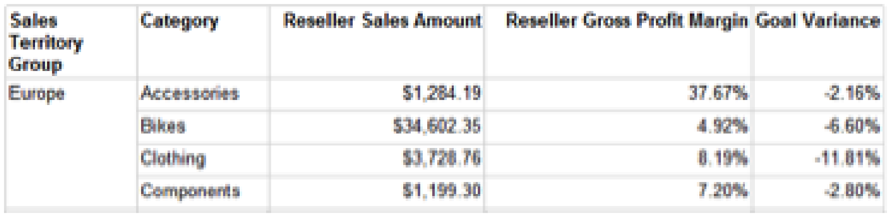
- **Excel**. This is a very common choice for browsing a cube since users are often already using Excel for other reasons and likely have some experience with pivot tables. Excel provides an easy-to-use interface to select dimensions for browsing, as shown in Figure 4, and also offers advanced functionality for filtering, sorting, and performing what-if analysis.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Product Categories | Bikes | | | |
| 2 | | | | | |
| 3 | Reseller Sales Amount | Column Labels | | | |
| 4 | Row Labels | Europe | North America | Pacific | Grand Total |
| 5 | Specialty Bike Shop | $791,996.83 | $4,989,397.35 | $298,723.55 | $6,080,117.73 |
| 6 | Value Added Reseller | $2,920,799.24 | $27,298,968.27 | $672,586.82 | $30,892,354.33 |
| 7 | Warehouse | $4,796,632.44 | $24,180,766.71 | $352,510.36 | $29,329,909.50 |
| 8 | Grand Total | $8,509,428.50 | $56,469,132.32 | $1,323,820.73 | $66,302,381.56 |

*Figure 4: Cube Browsing with Excel*

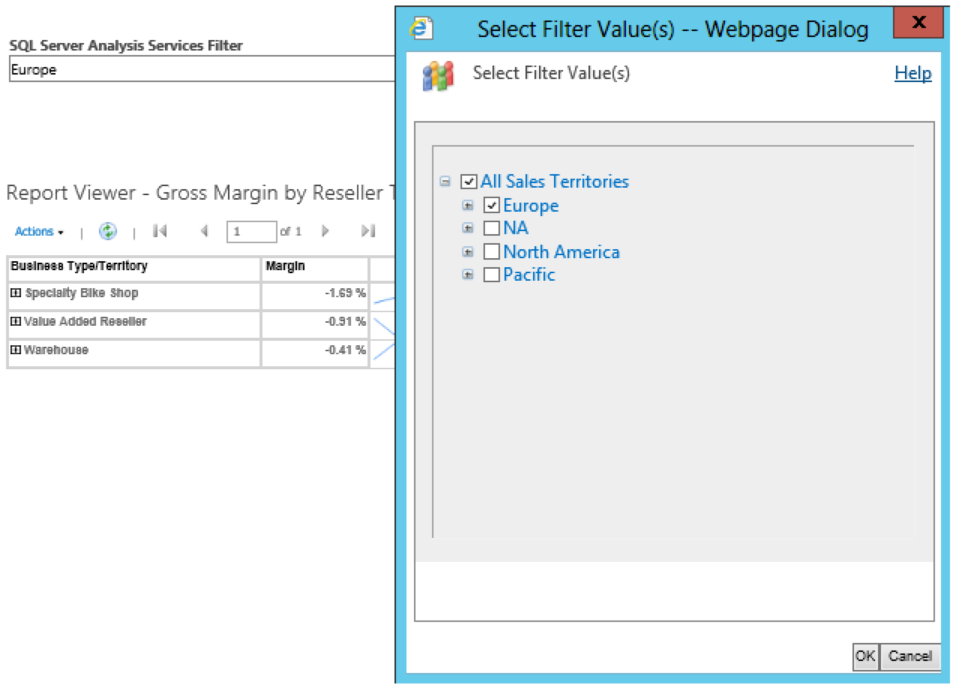
- **Reporting Services**. This is an option when users need to review information but are doing less exploration of the data. These users access the data by using pre-built static reports, as shown in Figure 5.



| Sales Territory Group | Category | Reseller Sales Amount | Reseller Gross Profit Margin | Goal Variance |
|---|---|---|---|---|
| Europe | Accessories | $1,284.19 | 37.67% | -2.16% |
| | Bikes | $34,602.35 | 4.92% | -6.60% |
| | Clothing | $3,728.76 | 8.19% | -11.81% |
| | Components | $1,199.30 | 7.20% | -2.80% |

*Figure 5: Report with Analysis Services Data Source*

- **SharePoint**. You can use Analysis Services as a data source for dashboard filters, as shown in Figure 6.



SQL Server Analysis Services Filter

Europe

Report Viewer - Gross Margin by Reseller T

| Business Type/Territory | Margin |
|---|---|
| Specialty Bike Shop | -1.69 % |
| Value Added Reseller | -0.91 % |
| Warehouse | -0.41 % |

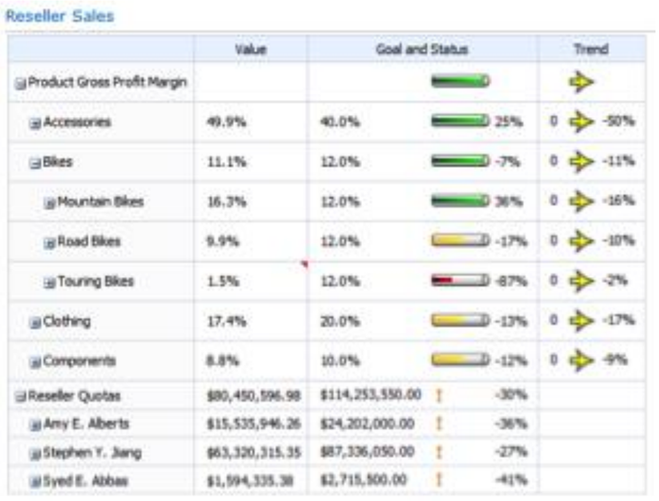
Select Filter Value(s) -- Webpage Dialog

Select Filter Value(s)

- [✓] All Sales Territories
  - [✓] Europe
  - [ ] NA
  - [ ] North America
  - [ ] Pacific

*Figure 6: Dashboard Filter with Analysis Services Data Source*

- **PerformancePoint Services**. You can create scorecards, as shown in Figure 7, and dashboards using Analysis Services as a data source.



Reseller Sales

| | Value | Goal and Status | | Trend | |
|---|---|---|---|---|---|
| Product Gross Profit Margin | | | | | |
| Accessories | 49.9% | 40.0% | 25% | 0 | -50% |
| Bikes | 11.1% | 12.0% | -7% | 0 | -11% |
| Mountain Bikes | 16.3% | 12.0% | 36% | 0 | -16% |
| Road Bikes | 9.9% | 12.0% | -17% | 0 | -10% |
| Touring Bikes | 1.5% | 12.0% | -87% | 0 | -2% |
| Clothing | 17.4% | 20.0% | -13% | 0 | -17% |
| Components | 8.8% | 10.0% | -12% | 0 | -9% |
| Reseller Quotas | $80,450,596.98 | $114,253,550.00 | -30% | | |
| Amy E. Alberts | $15,535,946.26 | $24,202,000.00 | -36% | | |
| Stephen Y. Jiang | $63,320,315.35 | $87,336,050.00 | -27% | | |
| Syed E. Abbas | $1,594,335.38 | $2,715,500.00 | -41% | | |

*Figure 7: Analysis Services Key Performance Indicators in a Scorecard*

## Database Development Process

Before diving into the details of Analysis Services database development, let's take a look at the general process:

1. Design a dimension model.
2. Develop dimension objects.
3. Develop cubes for the database.
4. Add calculations to the cube.
5. Deploy the database to the server.

First, you start by designing a **dimensional model**. You use either an existing dimensional model that you already have in a data mart or data warehouse, or you define the tables or views that you want to use as sources, set up logical primary keys, and define relationships to produce a structure that's very similar to a dimensional model that you would instantiate in the relational database. I describe this step in more detail in Chapter 2, "Designing the dimensional model."

Once the dimensional model is in place, you then work through the development of the **dimension objects**. When browsing a cube, you use dimensions to "slice and dice" the data. You will learn more about this step in the process in Chapter 3, "Developing dimensions."

The next step is to **develop one or more cubes** for the database. This is often an iterative process where you might go back and add more dimensions to the database and then return to do more development work on a cube. I will explain more about this in Chapter 4, "Developing cubes."

Eventually you **add calculations** to the cube to store the business logic in the cube for data that's not available in the raw data. There are specialized types of calculations to produce sets of dimension members and key performance indicators. You will learn how to work with all these types of calculations in Chapter 5, "Enhancing cubes with MDX."

During and after the development work, you **deploy the database** to the server and process objects to load them with data. It's not necessary to wait until you've completed each step in the development process to deploy. It's very common to develop a dimension, deploy it so that you can see the results, go back and modify the dimension, and then deploy again. You continue this cycle until you are satisfied with the dimension, and then you are ready to move on to the development of the next dimension.

## Anatomy of an Analysis Services Project

To start the multidimensional database development process in SSDT, you create a new project in SSDT. Here you can choose from one of the following project types:

- **Analysis Services Multidimensional and Data Mining Project**. You use this project type to build a project from scratch. The project will initially be empty, and then you build out each object individually, usually using wizards to get started quickly.
- **Import from Server (Multidimensional and Data Mining)**. If an Analysis Services database is already deployed to the server, you can import the database objects and have SSDT reverse engineer the design and create all the objects in the project.

Whether you start with an empty Analysis Services project or import objects from an existing Analysis Services database, there are several different types of project items that you have in an Analysis Services project:

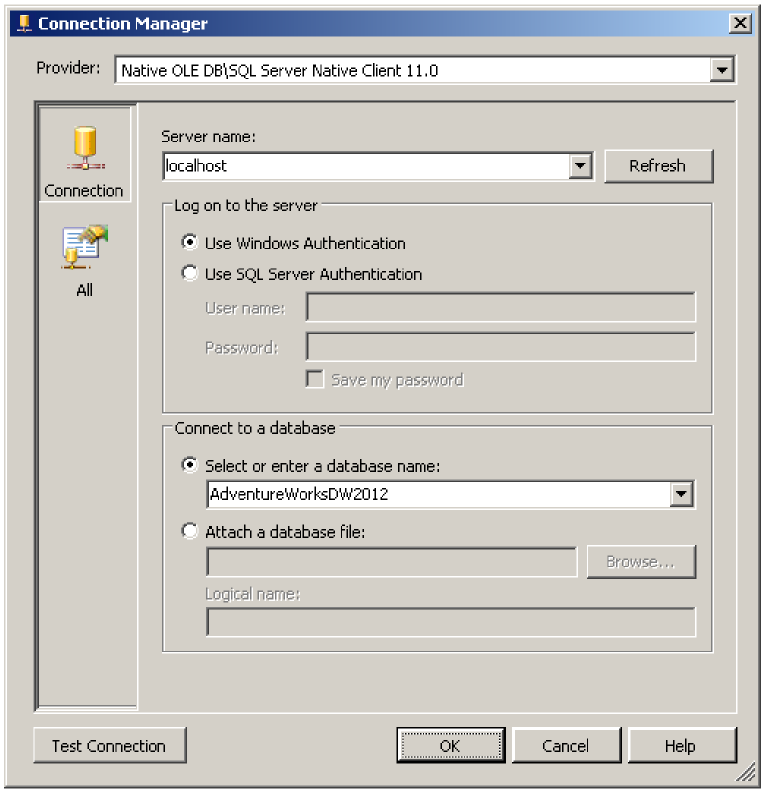
- **Data Source**. This item type defines how to connect to an OLE DB source that you want to use. If you need to change a server or database name, then you have only one place to make the change in SSDT.
- **Data Source View (DSV)**. The data source view represents the dimensional model. Everything you build into the Analysis Services database relies on the definitions of the data structures that you create in the data source view.
- **Cube**. An Analysis Services project has at least one cube file in it. You can create as many as you need.
- **Dimension**. Your project must have at least one dimension, although most cubes have multiple dimensions.
- **Role**. You use roles to configure user access permissions. I explain how to do this in Chapter 6, "Managing Analysis Services databases." It's not necessary to create a role in SSDT, however. You can add a role later in SSMS instead.

# Chapter 2 Working with the Data Source View

An Analysis Services multidimensional model requires you to use one or more relational data sources. Ideally, the data source is structured as a star schema, such as you typically find in a data warehouse or data mart. If not, you can make adjustments to a logical view of the data source to simulate a star schema. This logical view is known as a Data Source View (DSV) object in an Analysis Services database. In this chapter, I explain how to create a DSV and how to make adjustments to it in preparation for developing dimensions and cubes.

## Data Source

A DSV requires at least one data source, a file type in your Analysis Services project that defines the location of the data to load into the cube, the dimension objects in the database, and the information required to connect successfully to that data. You use a wizard to step through the process of creating this file. To launch the wizard, right-click the **Data Sources** folder in **Solution Explorer**. If you have an existing connection defined, you can select it in the list. Otherwise, click **New** to use the Connection Manager interface, shown in Figure 8, to select a provider, server, and database.



Provider: Native OLE DB\SQL Server Native Client 11.0

Server name: localhost

(●) Use Windows Authentication
( ) Use SQL Server Authentication
User name:
Password:
[ ] Save my password

(●) Select or enter a database name: AdventureWorksDW2012
( ) Attach a database file:
Logical name:

*Figure 8: Connection Manager for a New Data Source*

The provider you select can be a managed .NET provider, such as the SQL native client, when you're using SQL Server as the data source. You can also choose from several native OLE DB providers for other relational sources. Regardless, your data must be in a relational database. Analysis Services does not know how to retrieve data from Excel, applications like SAS, or flat files. You must first import data from those types of files into a database, and then you can use the data in Analysis Services.

After you select a provider, you then specify the server and database where the data is stored and also whether to use the Windows user or a database login for authentication whenever Analysis Services needs to connect to the data source. This process is similar to creating data sources in Integration Services or Reporting Services or other applications that require connections to data.

On the second page of the Data Source Wizard, you must define impersonation information. The purpose of the connection information in the Data Source file is to tell Analysis Services where to find data for the cubes and dimensions during processing. However, because processing is usually done on a scheduled basis, Analysis Services does not execute processing within the security context of a current user and requires impersonation information to supply a security context. There are four options:

- **Specific Windows user name and password**. You can hard-code a specific user name and password with this option.

- **Service account**. This is the account running the Analysis Services service, which is either a built-in account or a Windows account set up exclusively for the service. This might not be a good option if your data sources are on a remote server and you're using the Local Service or Local System accounts because those built-in accounts are restricted to the local server.

- **Current user's credentials**. You can select the option to use the credentials of the current user, but that's only useful when processing the database manually. Processing will fail if you set up a scheduled job through SQL Server Agent or an Integration Services task.

- **Inherit**. This option uses the database-level impersonation information (visible under **Management Studio** in the **Database Properties** dialog box). If the database-level impersonation is set to **Default**, Analysis Services uses the service account to make the connection. Otherwise, it uses the specified credentials.

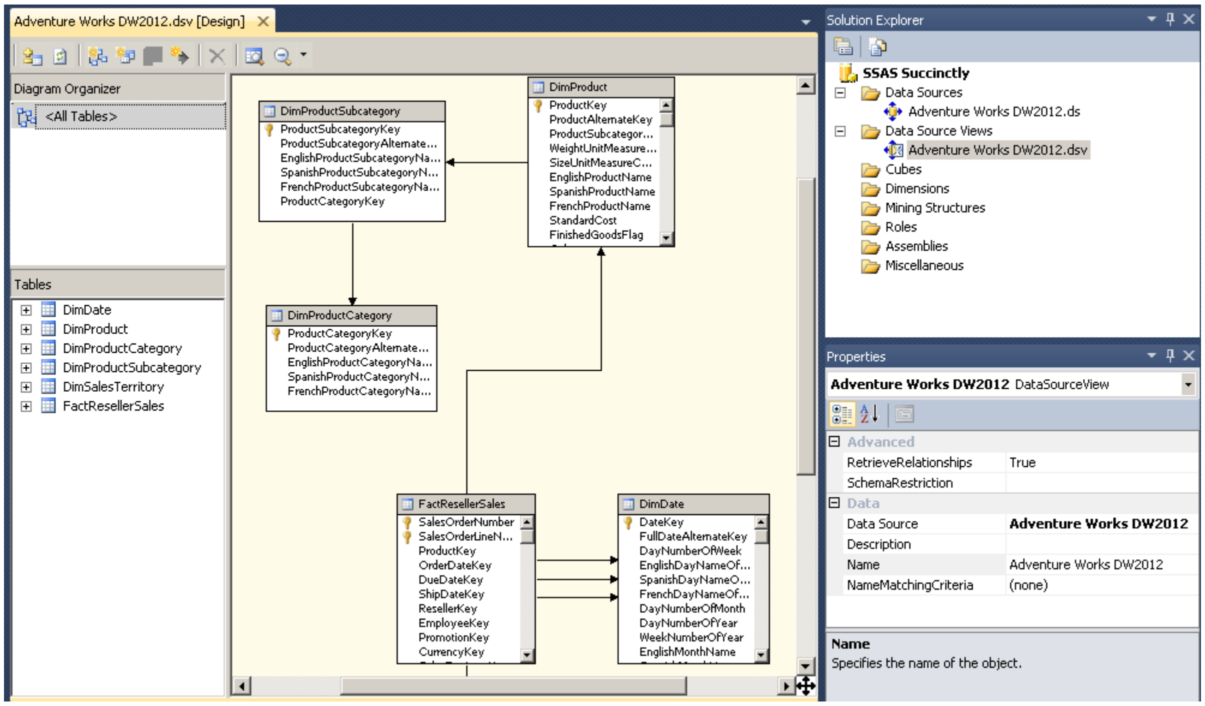
***Note: Regardless of the option you choose for impersonation, be sure the account has Read permissions on the data source. Otherwise, processing will fail.***

## Data Source View

The purpose of the DSV is to provide an abstraction layer between our physical sources in the relational database and the logical schema in SSAS. You can use it to combine multiple data sources that you might not be able to join together relationally, or to simulate structural changes that you wouldn't be allowed to make in the underlying source. Or you can use it to simplify a source that has a lot of tables so you can focus on only the tables needed to build the Analysis Services database. By having the metadata of the schema stored within the project, you can work on the Analysis Services database design when disconnected from the data source. Connectivity is required only when you're ready to load data into Analysis Services.

### Data Source View Wizard

The most common approach to building a DSV to is use existing tables in a data mart or data warehouse. These tables should already be populated with data. To start the Data Source View Wizard, right-click the **Data Source Views** folder in **Solution Explorer** and then select a data source. Select the tables or views that you want to use to develop dimensions and cubes. When you complete the wizard, your selections appear in diagram form in the center of the workspace and in tabular form on the left side of the workspace, as shown in Figure 9.



**Adventure Works DW2012** DataSourceView

| | |
|---|---|
| **Advanced** | |
| RetrieveRelationships | True |
| SchemaRestriction | |
| **Data** | |
| Data Source | **Adventure Works DW2012** |
| Description | |
| Name | Adventure Works DW2012 |
| NameMatchingCriteria | (none) |

**Name**
Specifies the name of the object.

*Figure 9: Data Source View*

## Primary Keys and Relationships

The tables in the DSV inherit the primary keys and foreign key relationships defined in the data source. You should see foreign key relationships between a fact table and related dimension tables, or between child levels in a snowflake dimension. Figure 9 includes examples of both types of relationships. The FactResellerSales table has foreign key relationships with two dimension tables, DimProduct and DimDate. In addition, foreign key relationships exist between levels in the product dimension. Specifically, these relationships appear between DimProductSubcategory and DimProductCategory, and between DimProduct and DimProductSubcategory.

One of the rules for dimension tables is that they must have a primary key. If for some reason your table doesn't have one, you can manufacture a logical primary key. Usually this situation arises during prototyping when you don't have a real data mart or data warehouse to use as a source. However, sometimes data warehouse developers leave off the primary key definition as a performance optimization for loading tables. To add a primary key, right-click the column containing values that uniquely identify each record in the table and select **Set Logical Primary Key** on the submenu. Your change does not update the physical schema in the database, but merely updates metadata about the table in the DSV.

Similarly, you should make sure that the proper relationships exist between fact and dimension tables. Sometimes these relationships are not created in the data source for performance reasons, or perhaps you are using tables from different data sources. Whatever the reason for the missing relationships, you can create logical relationships by dragging the foreign key column in one table to the primary key column in the other table. Take care to define the proper direction of a relationship. For example, the direction of the arrow needs to point away from the fact table and toward the dimension table, or away from a child level in a snowflake dimension and toward a parent level.

## Properties

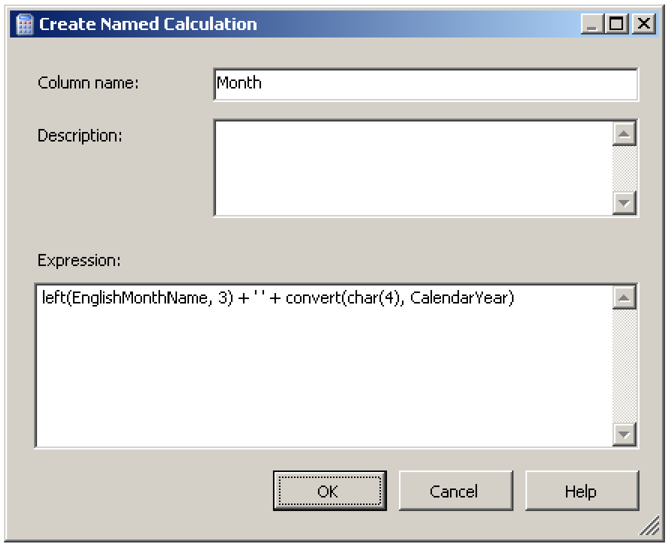
When you select a particular table or a column in a table, whether in the diagram or list of tables, you can view the related properties in the **Properties** window, which is displayed to the right of the diagram by default. You can change the names of tables or columns here if for some reason you don't have the necessary permissions to modify the names directly in the data source and want to provide friendlier names than might exist in the source. As you work with wizards during the development process, many objects inherit their names from the DSV. Therefore, the more work you do here to update the FriendlyName property, the easier your work will be during the later development tasks. For example, in a simple DSV in which I have the DimDate, DimProduct, DimSalesTerritory, and FactResellerSales tables, I change the FriendlyName property to Date, Product, Territory, and ResellerSales for each table, respectively.

## Named Calculations

A named calculation is simply an SQL expression that adds a column to a table in the DSV. You might do this when you have read-only access to a data source and need to adjust the data in some way. For example, you might want to concatenate two columns to produce a better report label for dimension items (known as members).

Like the other changes I've discussed in this chapter, the addition of a named calculation doesn't update the data source, but modifies the DSV only. The expression passes through directly to the underlying source, so we use the language that's applicable. For example, if SQL Server is your data source, you create a named calculation by using Transact-SQL syntax. There is no validation of our expression or expression builder in the dialog box. You must test the results elsewhere.

To add a named calculation, right-click the table and click **New Named Calculation** in the submenu. Then type an expression, as shown in Figure 10.



Column name: Month

Expression:

```
left(EnglishMonthName, 3) + ' ' + convert(char(4), CalendarYear)
```

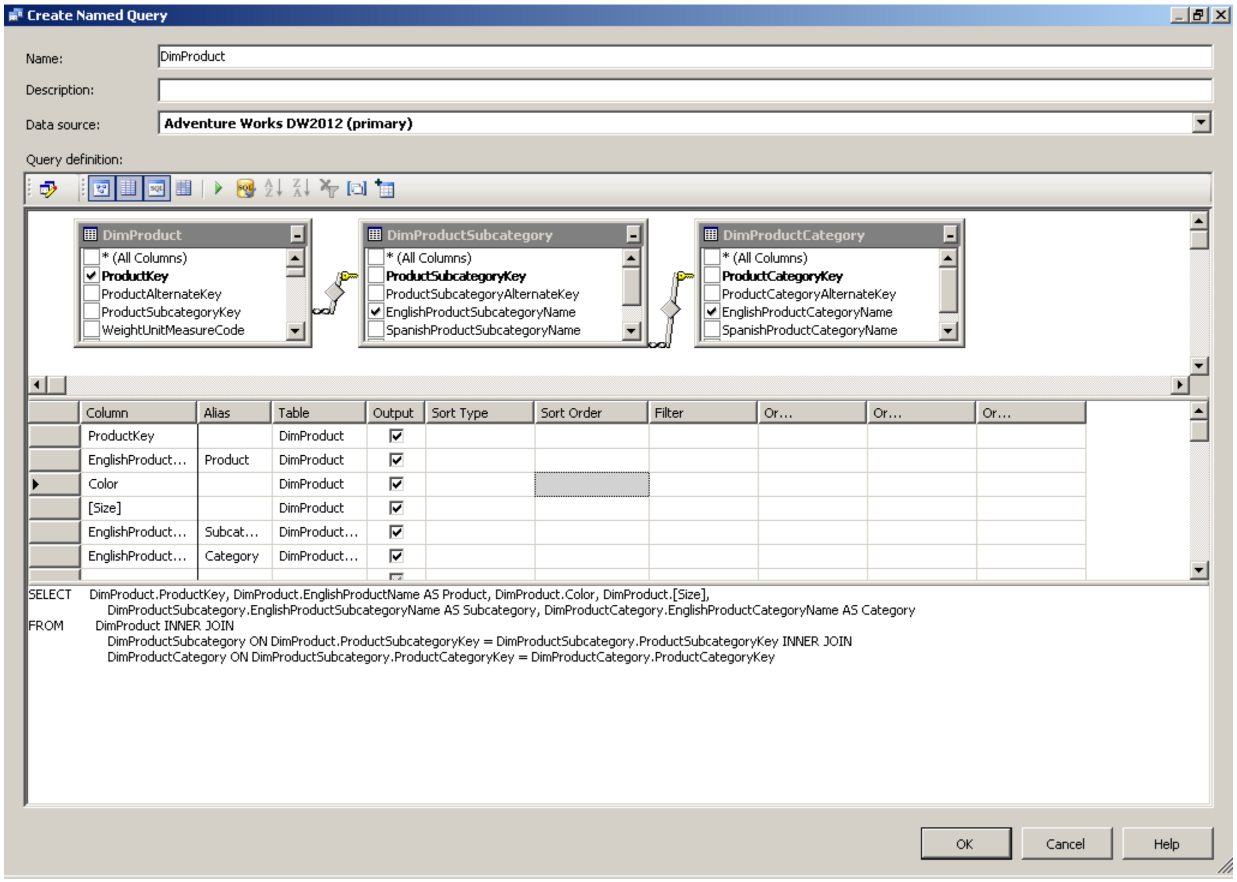
*Figure 10: Named Calculation*

After you add the expression as a named calculation, a new column is displayed in the DSV with a calculator icon. To test whether the expression is valid, right-click the table and select **Explore Data**. The expression is evaluated, allowing you to determine if you set up the expression correctly.

## Named Queries

When you need to do more than add a column to a table, you can use a named query instead of a named calculation. With a named query, you have complete control over the SELECT statement that returns data. It's just like creating a view in a relational database. One reason to do this is to eliminate columns from a table and thereby reduce its complexity. It's much easier for you to see the columns needed to build a dimension or cube when you can clear away the columns you don't need. Another reason is to add the equivalent of derived columns to a table. You can use an expression to add a new column to the table if you need to change the data in some way, like concatenating a first name and a last name together or multiplying a quantity sold by a price to get to the total sale amount for a transaction.

To create a named query, right-click an empty area in the DSV and select **New Named Query** or right-click on a table, point to **Replace Table**, and select **With New Named Query**. When you use SQL Server as a source for a named query, you have access to a graphical query builder interface as you design the query, as shown in Figure 11. You can also test the query inside the named query editor by clicking **Run** (the green arrow) in the toolbar.



Name: DimProduct

Description:

Data source: Adventure Works DW2012 (primary)

Query definition:

| Column | Alias | Table | Output | Sort Type | Sort Order | Filter | Or... | Or... | Or... |
|---|---|---|---|---|---|---|---|---|---|
| ProductKey | | DimProduct | ✓ | | | | | | |
| EnglishProduct... | Product | DimProduct | ✓ | | | | | | |
| Color | | DimProduct | ✓ | | | | | | |
| [Size] | | DimProduct | ✓ | | | | | | |
| EnglishProduct... | Subcat... | DimProduct... | ✓ | | | | | | |
| EnglishProduct... | Category | DimProduct... | ✓ | | | | | | |

```
SELECT    DimProduct.ProductKey, DimProduct.EnglishProductName AS Product, DimProduct.Color, DimProduct.[Size],
             DimProductSubcategory.EnglishProductSubcategoryName AS Subcategory, DimProductCategory.EnglishProductCategoryName AS Category
FROM       DimProduct INNER JOIN
             DimProductSubcategory ON DimProduct.ProductSubcategoryKey = DimProductSubcategory.ProductSubcategoryKey INNER JOIN
             DimProductCategory ON DimProductSubcategory.ProductCategoryKey = DimProductCategory.ProductCategoryKey
```

*Figure 11: Named Calculation*

# Chapter 3  Developing Dimensions

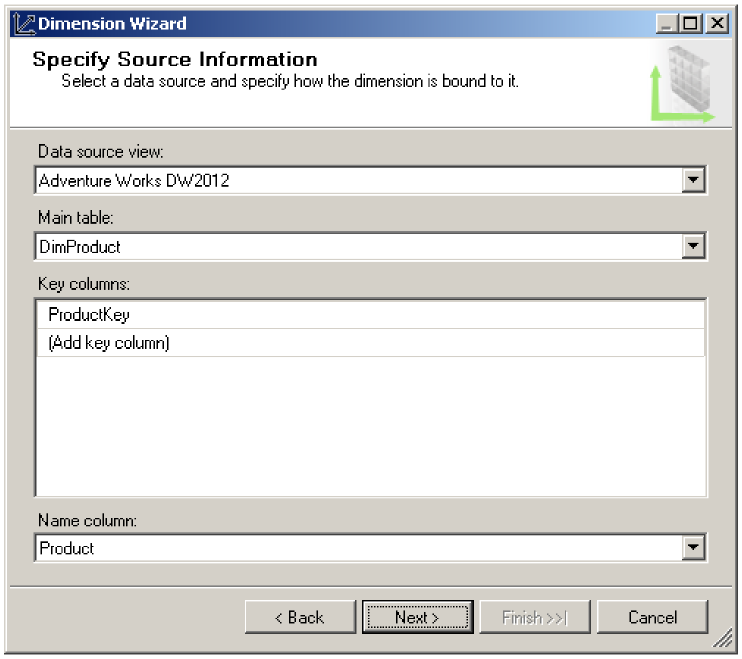
In this chapter, I explain the development tasks to perform during the development of dimensions for an Analysis Services database. You start by using the Dimension Wizard to build a dimension, and then configure attribute properties that control the content and the behavior of the dimension. You can also create hierarchies to facilitate drill-down and to optimize performance in cubes. If you need to support multiple languages in your cube, you can configure translations for the dimension. As you work, you might need to address best practice warnings that can help you avoid design and performance problems.

## Dimension Wizard

You use the Dimension Wizard to select the table to use as a source for the dimension and to set some initial properties for dimensions and attributes. To start the wizard, right-click the **Dimensions** folder in the Solution Explorer. On the **Select Creation Method** page of the wizard, keep the default selection of **Use An Existing Table**. The other options are used when you have no existing tables and want to generate a table that corresponds to a specific design.

***Note: The exception is the Generate a Time Table on the Server option, which creates and populates a dimension object only in the Analysis Services database without creating a corresponding table in the data source.***

On the **Specify Source Information** page of the wizard, shown in Figure 12, select the relevant table from the DSV, and identify the key column at minimum. The key column uniquely identifies each record in the dimension table and is usually the primary key of the table when you use a star schema for source data. You can specify multiple columns as key columns if the table has a composite key structure required to uniquely identify records.


Dimension Wizard

Specify Source Information

Select a data source and specify how the dimension is bound to it.

Data source view: Adventure Works DW2012

Main table: DimProduct

Key columns:

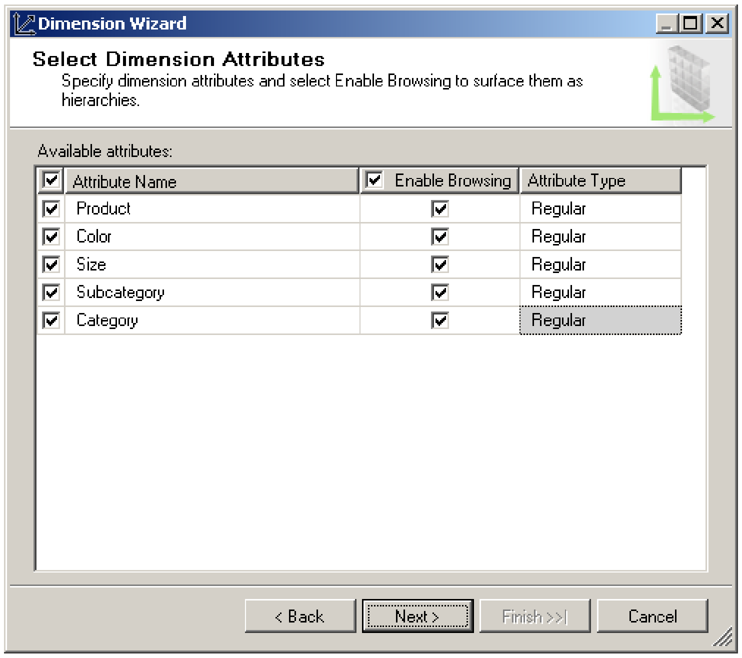
| |
|---|
| ProductKey |
| (Add key column) |

Name column: Product

*Figure 12:  Specify Source Information Page of Dimension Wizard*

Selecting a name column is optional in the Dimension Wizard. Its purpose is to display a label for dimension members when browsing the cube or its metadata. If you don't select a name column, the value in the key column is displayed instead.

On the **Select Dimension Attributes** page of the Dimension Wizard, shown in Figure 13, you pick the *attributes* to include in the dimension. Each attribute corresponds to a table column in the DSV. You can also rename the attributes on this page if you neglected to rename the column in the DSV.


Dimension Wizard

Select Dimension Attributes

Specify dimension attributes and select Enable Browsing to surface them as hierarchies.

Available attributes:

| ✓ | Attribute Name | ✓ Enable Browsing | Attribute Type |
|---|---|---|---|
| ✓ | Product | ✓ | Regular |
| ✓ | Color | ✓ | Regular |
| ✓ | Size | ✓ | Regular |
| ✓ | Subcategory | ✓ | Regular |
| ✓ | Category | ✓ | Regular |

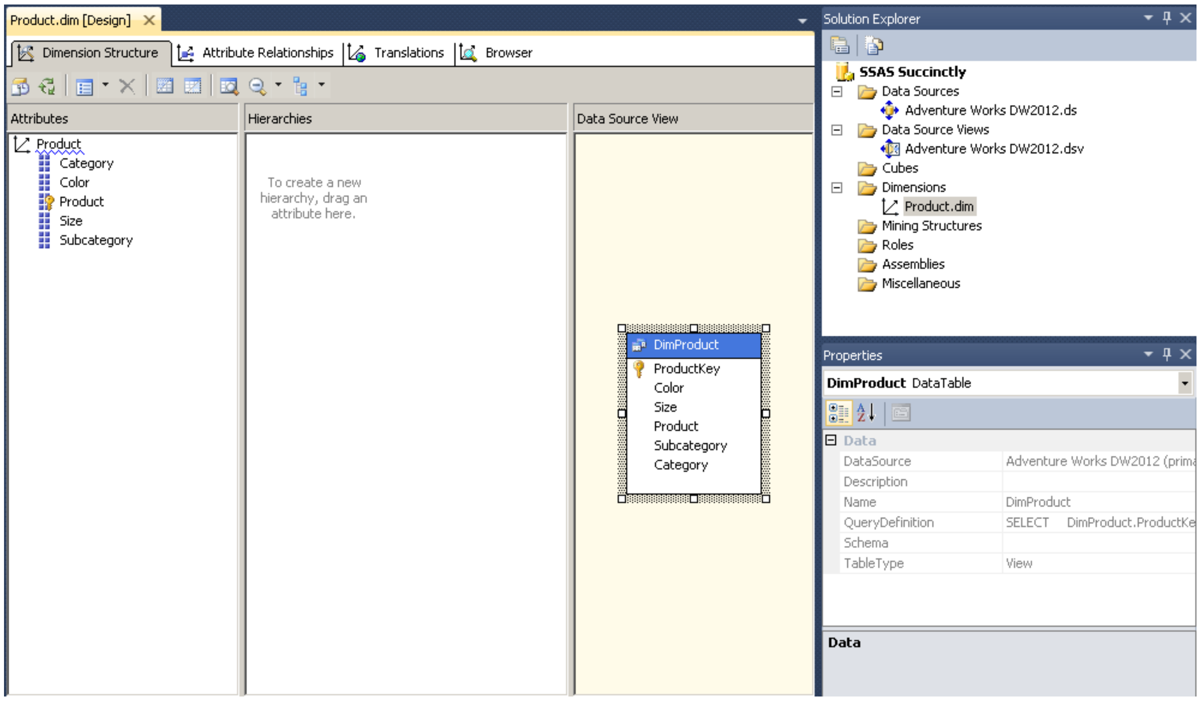
*Figure 13:  Select Dimension Attributes Page of Dimension Wizard*

Another task on this page is to specify whether users can view each attribute independently when browsing the cube. You do this by keeping the default selection in the **Enable Browsing** column. For example, if you have an attribute that is used exclusively for sorting purposes, you would clear the **Enable Browsing** check box for that attribute.

***Note: You can also set the Attribute Type on the Select Dimension Attributes page of the Dimension Wizard, but you might find it easier to perform this task by using the Business Intelligence Wizard accessible within the Dimension Designer instead.***

## Dimension Designer

After you complete the Dimension Wizard, the **Dimension Designer** opens for the newly created dimension, as shown in Figure 14. On the left side of the screen is the **Attributes** pane where you see a tree view of the dimension and its attributes. At the top of the tree, an icon with three arrows radiating from a common point identifies the dimension node. Below this node are several attributes corresponding to the selections in the Dimension Wizard. When you select an object in this tree view, the associated properties for the selected object are displayed in the **Properties** window, which appears in the lower right corner of the screen. Much of your work to fine-tune the dimension design involves configuring properties for the dimension and its attributes.



| | |
|---|---|
| DataSource | Adventure Works DW2012 (prima |
| Description | |
| Name | DimProduct |
| QueryDefinition | SELECT DimProduct.ProductKe |
| Schema | |
| TableType | View |

Figure 14: Dimension Designer

To the right of the Attributes pane is the **Hierarchies** pane. It is always initially empty because you must manually add each hierarchy, which is explained later in this chapter. The hierarchies that you add are called user-defined hierarchies to distinguish them from the attribute hierarchies. User-defined hierarchies can have multiple levels and are useful for helping users navigate from summarized to detailed information. By contrast, each attribute hierarchy contains an “All” level and a leaf level only. These levels are used for simple groupings when browsing data in a cube.

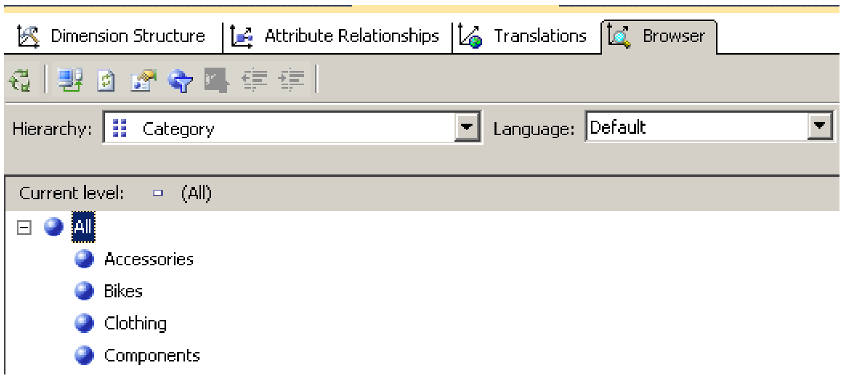
The third pane in the Dimension Designer is the **Data Source View** pane. Here you see a subset of the data source view, showing only the table or tables used to create the dimension.

***Tip: If you decide later to add more attributes to the dimension, you can drag each attribute from the Data Source View pane into the Attributes pane to add it to the dimension.***

The **Solution Explorer** in the top right corner always displays the files in your project. After you complete the Dimension Wizard, a new file with the DIM extensions is added to the project. If you later close the Dimension Designer for the current dimension, you can double-click the file name in the Solution Explorer to reopen it. As you add other dimensions to the project, each dimension has its own file in the Dimensions folder.

## Attributes

Attributes are always associated with one or more columns in the DSV. Let’s take a closer look at what this means by browsing an attribute in the **Browser**, which is the fourth tab of the Dimension Designer in SSDT. Before you can browse a dimension’s attributes, you must first deploy the project to send the dimension definition to the Analysis Services server and load the dimension object with data. To do this, click **Deploy** on the **Build** menu. After deployment is complete, you can select one attribute at a time in the **Hierarchy** drop-down list at the top of the page and then expand the **All** level to view the contents of the selected attribute, as shown in Figure 15.



Hierarchy: Category
Language: Default
Current level: (All)

All
Accessories
Bikes
Clothing
Components

*Figure 15: Dimension Designer*

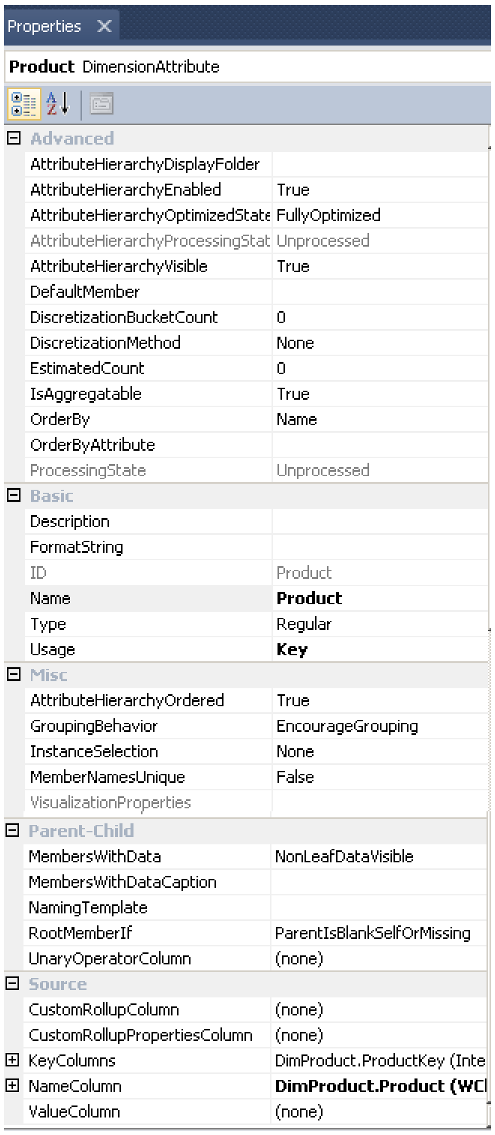
Each label in this list for the selected attribute is a *member*, including the **All** member at the top of the list. There is one member in the attribute for each distinct value in the key column that you specified in the Dimension Wizard, with the **All** member as the one exception. Analysis Services uses the **All** member to provide a grand total, or aggregated value, for the members when users browse the cube.

***Note: The number of members in the list (excluding the All member) is equal to the number of distinct values in the column specified as the attribute's Key Column property. The label displayed in the list is set by the column specified as the attribute's Name Column property.***

## Attribute Properties

Most of your work to set up a dimension involves configuring attribute properties. Let's take a closer look at that process now.

When you select an attribute in the **Attributes** pane, you can view its properties in the **Properties** window, as shown in Figure 16. In this example, you see the properties for the **Product** attribute, which is related to the primary key column in the Product table in the DSV. There are many properties available to configure here, but most of them are used for performance optimizations, special case design situations, or changing browsing behavior in a client tool.  You can tackle those properties on an as-needed basis. For the first phase of development, you should focus on the properties that you always want to double-check and reconfigure as needed. In particular, review the following properties: **KeyColumn**, **NameColumn**, **Name**, **OrderBy**, **AttributeHierarchyEnabled**, and **Usage**.



Properties

**Product** DimensionAttribute

| | |
|---|---|
| **Advanced** | |
| AttributeHierarchyDisplayFolder | |
| AttributeHierarchyEnabled | True |
| AttributeHierarchyOptimizedState | FullyOptimized |
| AttributeHierarchyProcessingStat | Unprocessed |
| AttributeHierarchyVisible | True |
| DefaultMember | |
| DiscretizationBucketCount | 0 |
| DiscretizationMethod | None |
| EstimatedCount | 0 |
| IsAggregatable | True |
| OrderBy | Name |
| OrderByAttribute | |
| ProcessingState | Unprocessed |
| **Basic** | |
| Description | |
| FormatString | |
| ID | Product |
| Name | **Product** |
| Type | Regular |
| Usage | **Key** |
| **Misc** | |
| AttributeHierarchyOrdered | True |
| GroupingBehavior | EncourageGrouping |
| InstanceSelection | None |
| MemberNamesUnique | False |
| VisualizationProperties | |
| **Parent-Child** | |
| MembersWithData | NonLeafDataVisible |
| MembersWithDataCaption | |
| NamingTemplate | |
| RootMemberIf | ParentIsBlankSelfOrMissing |
| UnaryOperatorColumn | (none) |
| **Source** | |
| CustomRollupColumn | (none) |
| CustomRollupPropertiesColumn | (none) |
| KeyColumns | DimProduct.ProductKey (Inte |
| NameColumn | **DimProduct.Product (WCl** |
| ValueColumn | (none) |

*Figure 16: Attribute Properties*

When you create a dimension using the Dimension Wizard, the **KeyColumn** and **NameColumn** properties are set in the wizard for the key attribute as described earlier in this chapter. You can use multiple key columns if necessary to uniquely identify an attribute member, but you can specify only one name column. If you need to use multiple columns to provide a name, you must go back to the DSV and concatenate the columns there by creating a named calculation or a named query to produce a single column with the value that you need.

You should next check the **Name** property of the attribute. It should be a user-friendly name. That is, it should be something users recognize and understand. It should be as short as possible, but still meaningful. For example, you might consider changing the **Name** property for the **Product** dimension from **Product Key** to **Product** to both shorten the name and eliminate confusion for users.

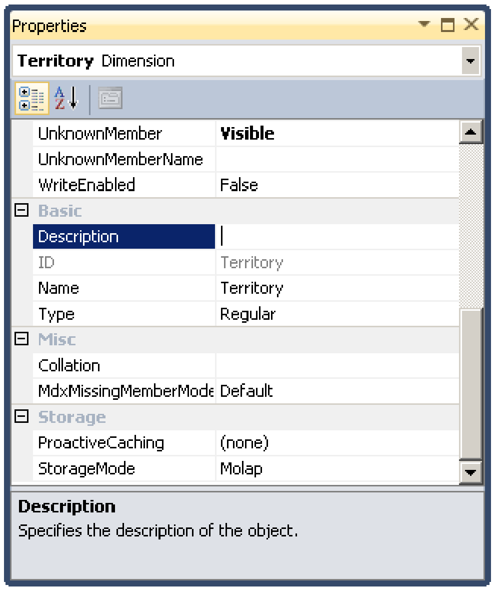
You might also need to change the sort order of members. The default sort order is by name, which means you see an alphabetical listing of members. However, sometimes you need a different sequence. For example, displaying month names in alphabetical order is usually not very helpful. You can order a list of attribute members by name, by key value, or even by some other attribute available in the same dimension. To do this, you must adjust the **OrderBy** property accordingly. If you choose to sort by an alternate attribute, you must also specify a value for the **OrderByAttribute** property.

Whenever you have attribute that you use just for sorting, you might not want it to be visible to users. When you want to use an attribute for sorting only, you can disable browsing altogether by setting the **AttributeHierarchyEnabled** property to **False**.

**Usage** is an important attribute property, but probably not one you need to change because it's usually auto-detected properly. There can be only one attribute that has a **Usage** value of **Key**, and that should be the attribute that is associated with the column identified as the primary key in the DSV. Otherwise, the value for this property should be **Regular**, unless you're working with a parent-child hierarchy, which I'll explain later in this chapter.

## Unknown Member

Sometimes the data source for a fact table's transaction or event data contains a null or invalid value in one of the foreign key columns for a dimension. By default, an attempt to process a cube associated with a fact table that has a data quality problem such as this results in an error. However, there might be a business case for which it is preferable to process the cube with a placeholder for the missing or invalid dimension reference. That way, the cube is processed successfully and all fact data is visible in the cube. To accommodate this scenario, enable the Unknown member for a dimension by selecting the dimension (the top node) in the **Attributes** pane and setting the **UnknownMember** property to **Visible**, as shown in Figure 17.



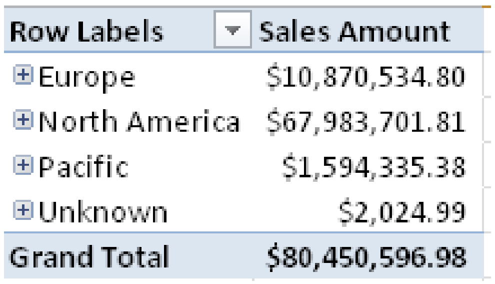
| | |
|---|---|
| UnknownMember | **Visible** |
| UnknownMemberName | |
| WriteEnabled | False |
| ⊟ Basic | |
| Description | |
| ID | Territory |
| Name | Territory |
| Type | Regular |
| ⊟ Misc | |
| Collation | |
| MdxMissingMemberMode | Default |
| ⊟ Storage | |
| ProactiveCaching | (none) |
| StorageMode | Molap |

**Description**
Specifies the description of the object.

*Figure 17: UnknownMember Property*

There are four possible values for the **UnknownMember** property:

- **Visible**. A new member labeled Unknown is displayed in the list of members for each attribute. An invalid fact record does not cause the cube processing to fail, and the fact record's measures are associated with the Unknown member, as shown in Figure 18.



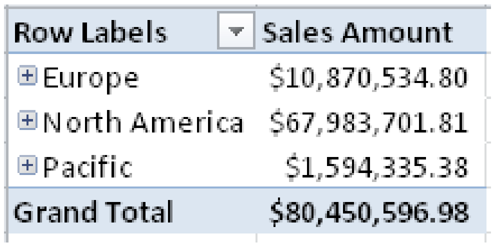
| Row Labels | Sales Amount |
|---|---|
| ⊞Europe | $10,870,534.80 |
| ⊞North America | $67,983,701.81 |
| ⊞Pacific | $1,594,335.38 |
| ⊞Unknown | $2,024.99 |
| **Grand Total** | **$80,450,596.98** |

*Figure 18: Invalid Fact Record Assigned to Visible Unknown Member*

***Note: If you decide to set the UnknownMember property to Visible, but prefer a different name, set the UnknownMemberName property for the dimension. For example, you might set the UnknownMemberName property to Unknown Territory. Because it is a dimension property, the same name appears in each attribute and user-defined hierarchy in the dimension.***

***Note: Configuring the UnknownMember property is not sufficient to ignore errors during cube processing if the dimension key is null or invalid. For more information regarding the options you have for managing error handling, see the Data Integrity Controls section of Handling Data Integrity Issues in Analysis Services 2005. Although this information was written for an earlier version of Analysis Services, it remains pertinent to later versions.***

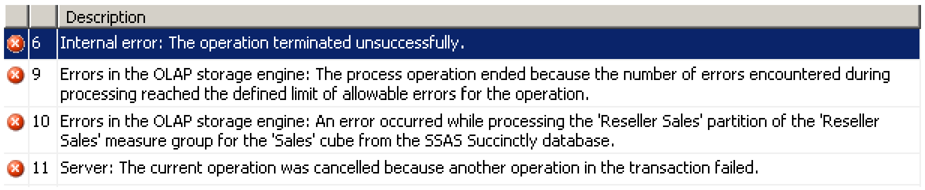
- **Hidden**. With this setting, an invalid fact record does not cause the cube process to fail. However, although the grand total for the dimension correctly displays the aggregate value for the associated measures, the Unknown member does not appear with other members when browsing the cube. Users might be confused when the values for visible members do not match the aggregated value, as shown in Figure 19.



| Row Labels | Sales Amount |
|---|---|
| ⊞Europe | $10,870,534.80 |
| ⊞North America | $67,983,701.81 |
| ⊞Pacific | $1,594,335.38 |
| **Grand Total** | **$80,450,596.98** |

*Figure 19: Invalid Fact Record Assigned to Hidden Unknown Member*

- **None**. An invalid fact record causes cube processing to fail. The problem must be resolved to complete cube processing successfully.



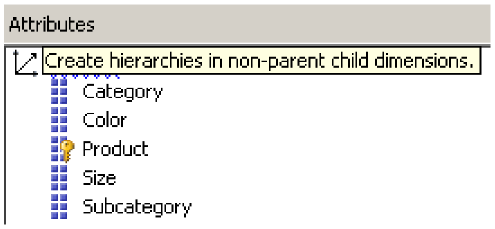
| | | Description |
|---|---|---|
| ✖ | 6 | Internal error: The operation terminated unsuccessfully. |
| ✖ | 9 | Errors in the OLAP storage engine: The process operation ended because the number of errors encountered during processing reached the defined limit of allowable errors for the operation. |
| ✖ | 10 | Errors in the OLAP storage engine: An error occurred while processing the 'Reseller Sales' partition of the 'Reseller Sales' measure group for the 'Sales' cube from the SSAS Succinctly database. |
| ✖ | 11 | Server: The current operation was cancelled because another operation in the transaction failed. |

*Figure 20: Processing Error Caused by Invalid Fact Record*

- **AutomaticNull**. This option applies only to Tabular mode in Analysis Services 2012.

## Design Warnings

You can avoid common problems in Analysis Services by reviewing and responding to design warnings that are displayed in the Dimension Designer and Cube Designer. A warning does not prevent the processing of a dimension or cube, but might result in less optimal performance or a less-than-ideal user experience if ignored. When you see a blue wavy underline in the designer, hover the pointer over the underscore to view the text of the warning, as shown in Figure 21.



Attributes

Create hierarchies in non-parent child dimensions.

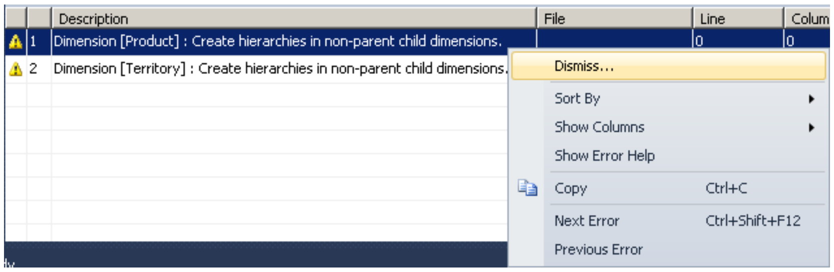
*Figure 21: Design Warning in the Attributes Pane of the Dimension Designer*

The steps you must perform to resolve the warning depend on the specific warning. Unfortunately, there is no guidance built into SSDT to help you. However, you can refer to Design Warning Rules at MSDN to locate the warning message and view the corresponding recommendation. For example, to resolve the warning in Figure 21, you add a user-defined hierarchy as described in the next section of this chapter. Sometimes the resolution you implement generates new warnings. Just continue to work through each warning until all warnings are cleared.

***Note: The Design Warning Rules link in the previous paragraph is specific to SQL Server 2008 R2, but is also applicable to SQL Server 2012 and SQL Server 2008.***

There might be circumstances in which you prefer to ignore a warning. For example, there may be times when you choose to leave attributes corresponding to hierarchy levels in a visible state rather than hide them according to the “Avoid visible attribute hierarchies for attributes used as levels in user-defined hierarchies” warning. You can do this to give greater flexibility to users for working with either attributes or user-defined hierarchies in a pivot table, rather than restricting them to the user-defined hierarchy only (but only after reviewing the options with the users). In that case, you can choose one of the following options:

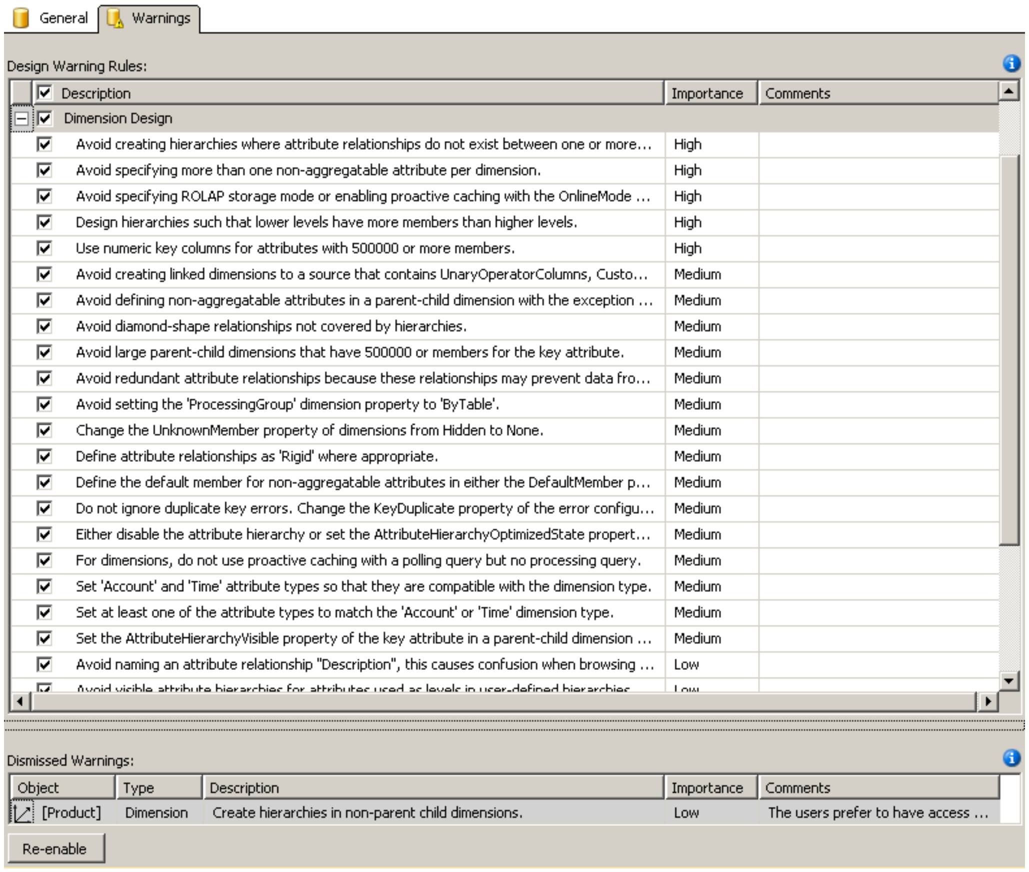
- **Dismiss a warning for an individual occurrence**. In the **Error List** window, which you can open from the **View** menu if it's not visible, right-click the best practice warning, and then select **Dismiss** on the submenu, as shown in Figure 22. In the **Dismiss Warning** dialog box, you have the option to enter a comment. After you dismiss the warning, the blue wavy underline disappears.



| | | Description | File | Line | Colum |
|---|---|---|---|---|---|
| ⚠ | 1 | Dimension [Product] : Create hierarchies in non-parent child dimensions. | | 0 | 0 |
| ⚠ | 2 | Dimension [Territory] : Create hierarchies in non-parent child dimensions. | | | |

*Figure 22: Dismissal of Best Practice Warning for a Dimension*

- **Dismiss a warning globally for all occurrences**. On the **Database** menu, select **Edit Database**, and then click the **Warnings** tab of the Database Designer, as shown in Figure 23. Here you can view warnings by type (such as Dimension Design), clear the check box to the left of a warning to dismiss it globally, and optionally type an explanation in the **Comments** column.



Design Warning Rules:

| ☑ Description | Importance | Comments |
|---|---|---|
| ☑ Dimension Design | | |
| ☑ Avoid creating hierarchies where attribute relationships do not exist between one or more... | High | |
| ☑ Avoid specifying more than one non-aggregatable attribute per dimension. | High | |
| ☑ Avoid specifying ROLAP storage mode or enabling proactive caching with the OnlineMode ... | High | |
| ☑ Design hierarchies such that lower levels have more members than higher levels. | High | |
| ☑ Use numeric key columns for attributes with 500000 or more members. | High | |
| ☑ Avoid creating linked dimensions to a source that contains UnaryOperatorColumns, Custo... | Medium | |
| ☑ Avoid defining non-aggregatable attributes in a parent-child dimension with the exception ... | Medium | |
| ☑ Avoid diamond-shape relationships not covered by hierarchies. | Medium | |
| ☑ Avoid large parent-child dimensions that have 500000 or members for the key attribute. | Medium | |
| ☑ Avoid redundant attribute relationships because these relationships may prevent data fro... | Medium | |
| ☑ Avoid setting the 'ProcessingGroup' dimension property to 'ByTable'. | Medium | |
| ☑ Change the UnknownMember property of dimensions from Hidden to None. | Medium | |
| ☑ Define attribute relationships as 'Rigid' where appropriate. | Medium | |
| ☑ Define the default member for non-aggregatable attributes in either the DefaultMember p... | Medium | |
| ☑ Do not ignore duplicate key errors. Change the KeyDuplicate property of the error configu... | Medium | |
| ☑ Either disable the attribute hierarchy or set the AttributeHierarchyOptimizedState propert... | Medium | |
| ☑ For dimensions, do not use proactive caching with a polling query but no processing query. | Medium | |
| ☑ Set 'Account' and 'Time' attribute types so that they are compatible with the dimension type. | Medium | |
| ☑ Set at least one of the attribute types to match the 'Account' or 'Time' dimension type. | Medium | |
| ☑ Set the AttributeHierarchyVisible property of the key attribute in a parent-child dimension ... | Medium | |
| ☑ Avoid naming an attribute relationship "Description", this causes confusion when browsing ... | Low | |
| ☑ Avoid visible attribute hierarchies for attributes used as levels in user-defined hierarchies | Low | |

Dismissed Warnings:

| Object | Type | Description | Importance | Comments |
|---|---|---|---|---|
| [Product] | Dimension | Create hierarchies in non-parent child dimensions. | Low | The users prefer to have access ... |

*Figure 23: Global and Dismissed Best Practice Warnings*

***Tip: If you want to enable a previously dismissed instance warning, select it in the Dismissed Warnings list at the bottom of the Warnings page of the Database Designer, and then click Re-enable.***

## User-Defined Hierarchies

Users can always arrange attributes any way they like in a cube browser, but it's usually helpful to add a user-defined hierarchy for them to use. User-defined hierarchies are never automatically detected; you must add them manually. This type of hierarchy structure is called a user-defined hierarchy because as an Analysis Services developer, you are defining the hierarchy, in contrast to automatic generation of an attribute hierarchy by Analysis Services when you add an attribute to a dimension.

To create a hierarchy, drag attributes to the **Hierarchies** pane in the Dimension Designer. As you add each attribute, place it above or below an existing level to achieve the desired hierarchical order, as shown in Figure 24. You should also rename the hierarchy to provide a more user-friendly label.

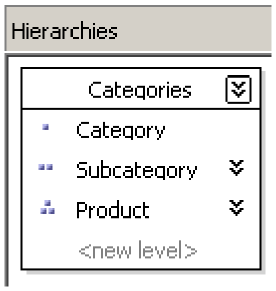
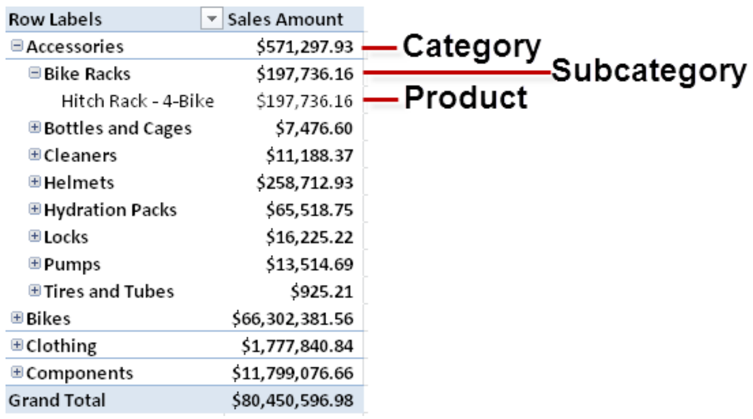


*Figure 24: User-Defined Hierarchy*

When the user selects a hierarchy in a browser, the user can drill easily from one level to the next. For example, in an Excel pivot table, the user can expand a member at the Category level to show members on the Subcategory level. Then, the user can select a member of the Subcategory level to see members in the Product level, as shown in Figure 25.



| Row Labels | Sales Amount |
|---|---|
| ⊟Accessories | $571,297.93 |
| ⊟Bike Racks | $197,736.16 |
| Hitch Rack - 4-Bike | $197,736.16 |
| ⊞Bottles and Cages | $7,476.60 |
| ⊞Cleaners | $11,188.37 |
| ⊞Helmets | $258,712.93 |
| ⊞Hydration Packs | $65,518.75 |
| ⊞Locks | $16,225.22 |
| ⊞Pumps | $13,514.69 |
| ⊞Tires and Tubes | $925.21 |
| ⊞Bikes | $66,302,381.56 |
| ⊞Clothing | $1,777,840.84 |
| ⊞Components | $11,799,076.66 |
| Grand Total | $80,450,596.98 |

*Figure 25: Members in a User-Defined Hierarchy*

## Natural Hierarchies

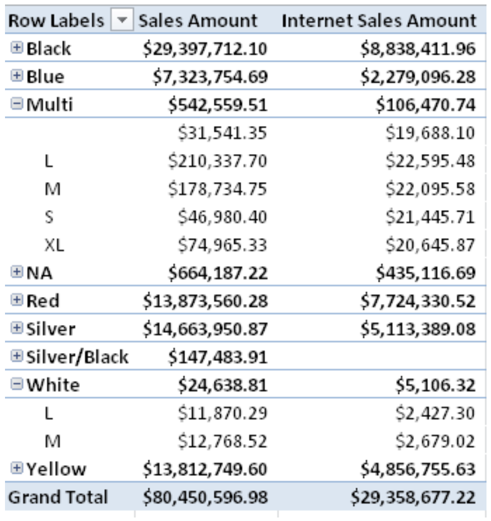
The addition of a hierarchy to a dimension not only helps users navigate data more efficiently from summary to detail data, but it can also improve query performance when a hierarchy contains a natural one-to-many relationship between each level in the hierarchy from top to bottom, such as exists between Category, Subcategory, and Product. This type of structure is commonly known as a natural hierarchy.

When a natural hierarchy exists between levels, Analysis Services can store data more efficiently and can also build aggregations to pre-compute data in the cube. When a user asks for sales by category, for example, the server doesn't have to scan through each transaction first and then group by category. Instead, the category totals are available either directly or indirectly, and the query results return from Analysis Services much faster than they would if Analysis Services were required to calculate the sum based on the data at the transaction level. I explain more about aggregations in Chapter 4, "Developing cubes."

***Tip: You might decide to allow users to access an attribute exclusively from within a hierarchy. This is useful particularly when you have a very large set of members in an attribute, such as customers. In that case, it's usually preferable to require the users to start by adding a hierarchy to the pivot table and then filtering down to a smaller set, such as customers in a particular city. To do this, set the AttributeHierarchyVisible property to false for each attribute. The attribute will be visible within a user-defined hierarchy, but will not appear in the dimension's list of attributes as an independent attribute hierarchy.***

## Unnatural Hierarchies

You can also create an unnatural hierarchy in Analysis Services. The purpose of an unnatural hierarchy is to provide a predefined grouping of attributes. For example, in the **Product** dimension, you might have users who frequently analyze product sales by color and by size. You can set up a hierarchy with the color and size attributes, and then users can use this hierarchy in the browser to drill from color to size, as shown in Figure 26. In a natural hierarchy, a member in a lower level can be associated with only one member in its parent level, but in an unnatural hierarchy, users see sizes like L and M associated with both the Multi and White colors.



| Row Labels | Sales Amount | Internet Sales Amount |
|---|---|---|
| ⊞Black | $29,397,712.10 | $8,838,411.96 |
| ⊞Blue | $7,323,754.69 | $2,279,096.28 |
| ⊟Multi | $542,559.51 | $106,470.74 |
| | $31,541.35 | $19,688.10 |
| L | $210,337.70 | $22,595.48 |
| M | $178,734.75 | $22,095.58 |
| S | $46,980.40 | $21,445.71 |
| XL | $74,965.33 | $20,645.87 |
| ⊞NA | $664,187.22 | $435,116.69 |
| ⊞Red | $13,873,560.28 | $7,724,330.52 |
| ⊞Silver | $14,663,950.87 | $5,113,389.08 |
| ⊞Silver/Black | $147,483.91 | |
| ⊟White | $24,638.81 | $5,106.32 |
| L | $11,870.29 | $2,427.30 |
| M | $12,768.52 | $2,679.02 |
| ⊞Yellow | $13,812,749.60 | $4,856,755.63 |
| Grand Total | $80,450,596.98 | $29,358,677.22 |

*Figure 26:  Unnatural Hierarchy*

In an unnatural hierarchy, there is no query performance benefit. It’s simply a convenience for common groupings users work with frequently and completely optional.

## Attribute Relationships

When you have a user-defined hierarchy in a dimension, it’s important to properly define attribute relationships. Attribute relationships are used to ensure that aggregations work efficiently and totals are calculated correctly. When you first create a hierarchy commonly found in the Date dimension, each upper level of the hierarchy has a direct relationship with the dimension’s key attribute. You can review these relationships on the Attribute Relationships tab of the Dimension Designer as shown in Figure 27.

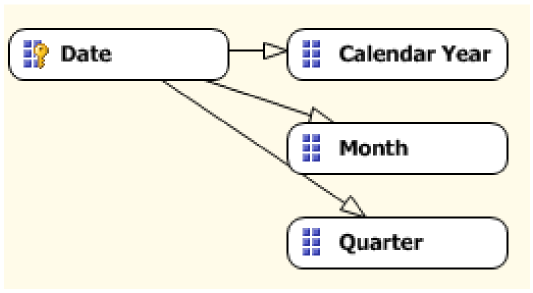


*Figure 27:  Default Attribute Relationships*

In some cases, if attribute relationships are not defined correctly, it's possible for totals in the cube to be calculated incorrectly. However, a greater risk is the introduction of a potential performance bottleneck. For example, let's say that aggregations are available for **Month**, but not for **Quarter** or for **Year**. When a user requests sales by quarter, Analysis Services must use the transaction-level data in the fact table to calculate the sales by quarter. On the other hand, if proper relationships exist between attributes, Analysis Services uses values already available for lower-level attributes to compute totals for higher-level attributes and usually calculates these values much faster. Even without aggregations, query performance benefits from attribute relationships because they help Analysis Services narrow down the amount of cube space that has to be scanned in order to retrieve results for a query.

To correct attribute relationships on the Attribute Relationships tab of the Dimension Designer, drag a lower-level attribute to the attribute on the level above it. For example, drag **Month** to **Quarter**, and then **Quarter** to **Calendar Year**. Typically, you don't need to delete an erroneous relationship first, but if necessary you can select the arrow representing the relationship between two attributes and press Delete to remove it. A correctly defined set of attribute relationships for the Date dimension is shown in Figure 28.

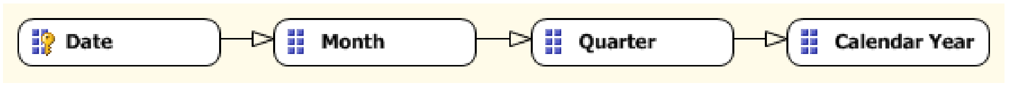


*Figure 28: Correct Attribute Relationships*

Although the fact table in the data source stores a date, Analysis Services can calculate month, quarter, or year by rolling up, or aggregating, values by following the chain of attribute relationships from left to right. Attribute relationships represent many-to-one relationships moving from left to right. In other words, there are many dates associated with a single month, and many months associated with a single quarter, and many quarters for a single year.

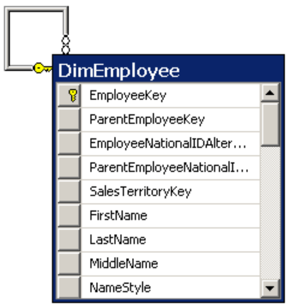
Attribute relationships can also have either flexible or rigid relationship types, each of which has a different effect on dimension processing. By default, an attribute relationship type is flexible, as indicated by a white arrowhead. To change the relationship type, right-click the arrow between two attributes, point to **Relationship Type**, and select one of the following relationship types, as applicable:

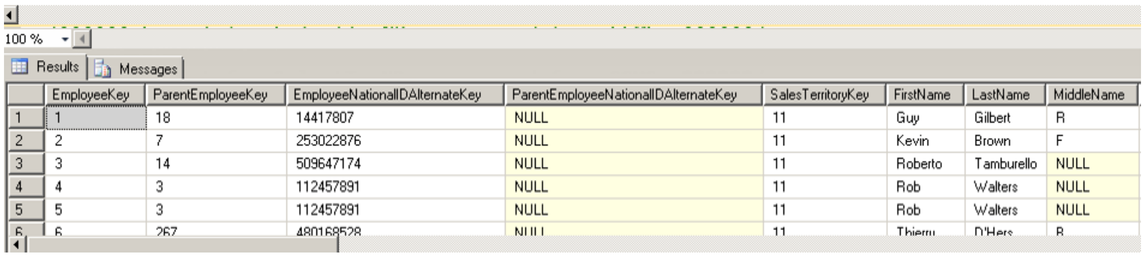
- **Flexible**. A flexible attribute relationship type allows you to update your source dimension table by reassigning a member from one parent to another. For example, let's say you decide to break the Bikes category down into two categories: Road Bikes and Off-Road Bikes, and you assign the Mountain Bikes subcategory to Off-Road Bikes, add a new Cyclocross subcategory to Off-Road Bikes, and assign Road Bikes and Touring Bikes to the Road Bikes category. When the Category and Subcategory attributes have a flexible relationship type, you can make this type of change to reassign members from one parent (Bikes) to another (Off-Road Bikes) easily by processing the dimension as an update as described in Chapter 6, "Managing Analysis Services databases." An update process does not require you to process the cube, which means the processing occurs very quickly. The downside of this approach is that any aggregations that were built are removed from the database, which in turn means queries slow down until you can take the time to process aggregations later.

- **Rigid**. A rigid relationship type allows you to perform an update process on the dimension without losing aggregations. However, if the dimension table is updated to reflect a change where a member moves from one parent to another, an error is raised and stops processing. Therefore, set rigid relationships only when you know that data is not likely to be rearranged later. For example, in a Date dimension, January 1 always has the month of January as a parent and will not be assigned later to a different month, such as August.

## Parent-Child Hierarchy

Another type of hierarchy that you might need to create in Analysis Services is a parent-child hierarchy.  In this type of hierarchy, the levels of the hierarchy come from the built-in relationships found in a self-referencing table, like the **DimEmployee** table in the AdventureWorksDW2012 database. In this table, the **ParentEmployeeKey** column has a foreign key relationship to the primary key column, **EmployeeKey**, as shown in Figure 29.





| | EmployeeKey | ParentEmployeeKey | EmployeeNationalIDAlternateKey | ParentEmployeeNationalIDAlternateKey | SalesTerritoryKey | FirstName | LastName | MiddleName |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 18 | 14417807 | NULL | 11 | Guy | Gilbert | R |
| 2 | 2 | 7 | 253022876 | NULL | 11 | Kevin | Brown | F |
| 3 | 3 | 14 | 509647174 | NULL | 11 | Roberto | Tamburello | NULL |
| 4 | 4 | 3 | 112457891 | NULL | 11 | Rob | Walters | NULL |
| 5 | 5 | 3 | 112457891 | NULL | 11 | Rob | Walters | NULL |
| 6 | 6 | 267 | 480168528 | NULL | 11 | Thierry | D'Hers | B |

*Figure 29:  Correct Attribute Relationships*

The benefit of this structure is that there is no need to know in advance how many levels exist in the hierarchy, nor must you create each level explicitly in the dimension. This type of structure is very flexible and used most often for organizational charts, a chart of accounts in a financial application, or a bill of materials list in a manufacturing application. The self-referencing join in the table is automatically detected in the DSV and an attribute with the foreign key relationship is displayed in the Attribute pane with a special icon, as shown in Figure 30, to indicate its **Usage** property is set to **Parent**.

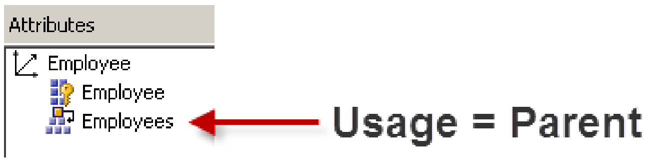
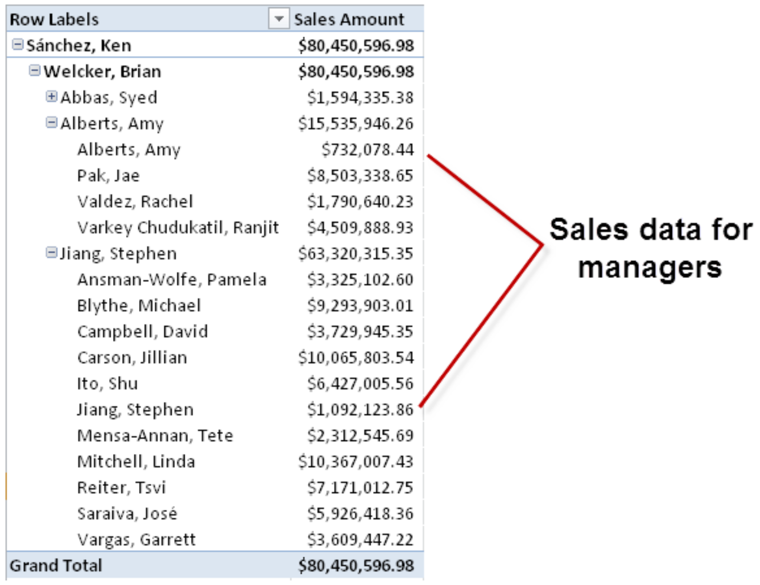


*Figure 30:  Parent Attribute*

***Note: Each dimension can have only one attribute with the Usage property set to Parent.***

Another benefit of a parent-child hierarchy is the ability to store data in the fact table for non-leaf members. For example, in the **FactResellerSales** table, there are sales transactions for sales people like Jae Pak and Linda Mitchell, but there are also transactions associated with their respective managers, Amy Alberts, and Stephen Jiang.

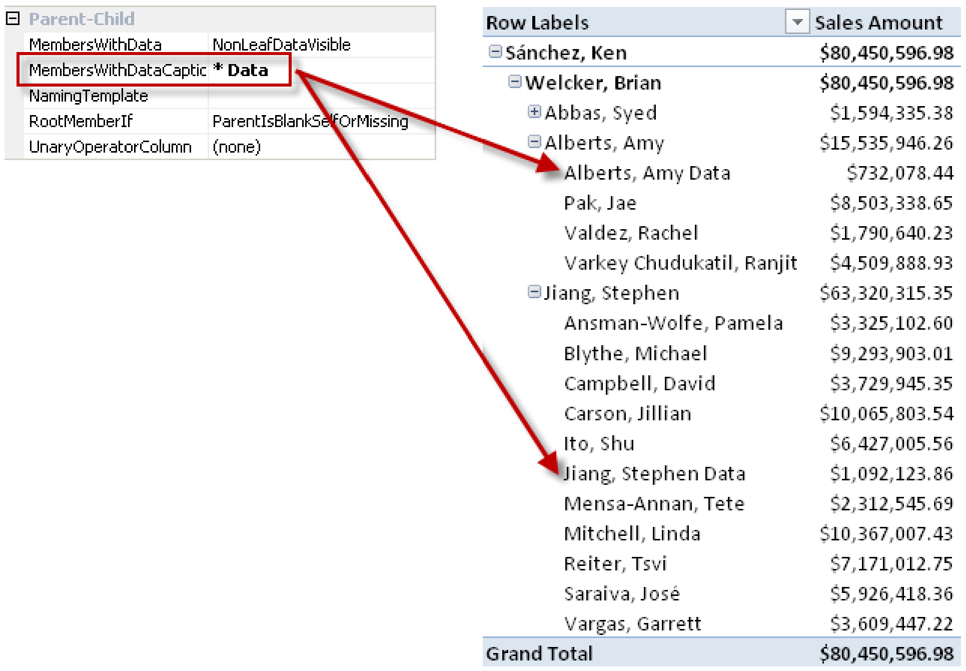
You use the parent attribute's **MembersWithData** property to control whether the fact table data for the managers is visible. By default, this property is set to **True**, which means the sales data for managers is visible in the cube, as shown in Figure 31.



| Row Labels | Sales Amount |
|---|---|
| ⊟Sánchez, Ken | $80,450,596.98 |
| ⊟Welcker, Brian | $80,450,596.98 |
| ⊞Abbas, Syed | $1,594,335.38 |
| ⊟Alberts, Amy | $15,535,946.26 |
| Alberts, Amy | $732,078.44 |
| Pak, Jae | $8,503,338.65 |
| Valdez, Rachel | $1,790,640.23 |
| Varkey Chudukatil, Ranjit | $4,509,888.93 |
| ⊟Jiang, Stephen | $63,320,315.35 |
| Ansman-Wolfe, Pamela | $3,325,102.60 |
| Blythe, Michael | $9,293,903.01 |
| Campbell, David | $3,729,945.35 |
| Carson, Jillian | $10,065,803.54 |
| Ito, Shu | $6,427,005.56 |
| Jiang, Stephen | $1,092,123.86 |
| Mensa-Annan, Tete | $2,312,545.69 |
| Mitchell, Linda | $10,367,007.43 |
| Reiter, Tsvi | $7,171,012.75 |
| Saraiva, José | $5,926,418.36 |
| Vargas, Garrett | $3,609,447.22 |
| Grand Total | $80,450,596.98 |

*Figure 31: Parent Members with Data*

You can use the **MembersWithDataCaption** property to make the distinction between the managers' sales and the subtotal of sales for the managers' employees and the manager. Use an asterisk as a placeholder for the member name and append a string to include in the caption, as shown in Figure 32.

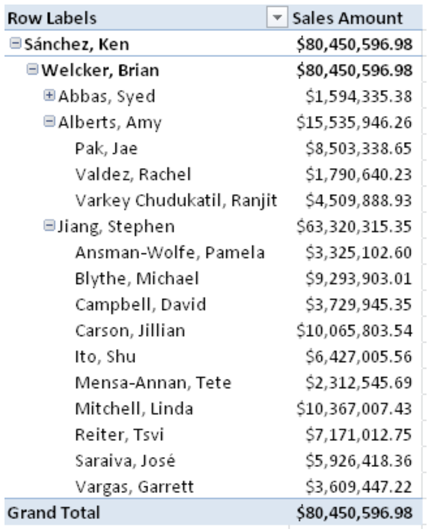


| Parent-Child | |
|---|---|
| MembersWithData | NonLeafDataVisible |
| MembersWithDataCaptic | * Data |
| NamingTemplate | |
| RootMemberIf | ParentIsBlankSelfOrMissing |
| UnaryOperatorColumn | (none) |

| Row Labels | Sales Amount |
|---|---|
| ⊟Sánchez, Ken | $80,450,596.98 |
| ⊟Welcker, Brian | $80,450,596.98 |
| ⊞Abbas, Syed | $1,594,335.38 |
| ⊟Alberts, Amy | $15,535,946.26 |
| Alberts, Amy Data | $732,078.44 |
| Pak, Jae | $8,503,338.65 |
| Valdez, Rachel | $1,790,640.23 |
| Varkey Chudukatil, Ranjit | $4,509,888.93 |
| ⊟Jiang, Stephen | $63,320,315.35 |
| Ansman-Wolfe, Pamela | $3,325,102.60 |
| Blythe, Michael | $9,293,903.01 |
| Campbell, David | $3,729,945.35 |
| Carson, Jillian | $10,065,803.54 |
| Ito, Shu | $6,427,005.56 |
| Jiang, Stephen Data | $1,092,123.86 |
| Mensa-Annan, Tete | $2,312,545.69 |
| Mitchell, Linda | $10,367,007.43 |
| Reiter, Tsvi | $7,171,012.75 |
| Saraiva, José | $5,926,418.36 |
| Vargas, Garrett | $3,609,447.22 |
| Grand Total | $80,450,596.98 |

*Figure 32: Captions for Parent Members with Data*

An alternate solution is to hide the data by changing the **MembersWithData** property to **NonLeafDataHidden**. In that case, though, the subtotals by manager are still correct and the manager's sales can be inferred, as shown in Figure 33. Therefore, this technique is not a security measure, but simply an alternate way to represent data when fact records exist for non-leaf dimension members.



| Row Labels | Sales Amount |
|---|---|
| ⊟Sánchez, Ken | $80,450,596.98 |
| ⊟Welcker, Brian | $80,450,596.98 |
| ⊞Abbas, Syed | $1,594,335.38 |
| ⊟Alberts, Amy | $15,535,946.26 |
| Pak, Jae | $8,503,338.65 |
| Valdez, Rachel | $1,790,640.23 |
| Varkey Chudukatil, Ranjit | $4,509,888.93 |
| ⊟Jiang, Stephen | $63,320,315.35 |
| Ansman-Wolfe, Pamela | $3,325,102.60 |
| Blythe, Michael | $9,293,903.01 |
| Campbell, David | $3,729,945.35 |
| Carson, Jillian | $10,065,803.54 |
| Ito, Shu | $6,427,005.56 |
| Mensa-Annan, Tete | $2,312,545.69 |
| Mitchell, Linda | $10,367,007.43 |
| Reiter, Tsvi | $7,171,012.75 |
| Saraiva, José | $5,926,418.36 |
| Vargas, Garrett | $3,609,447.22 |
| Grand Total | $80,450,596.98 |

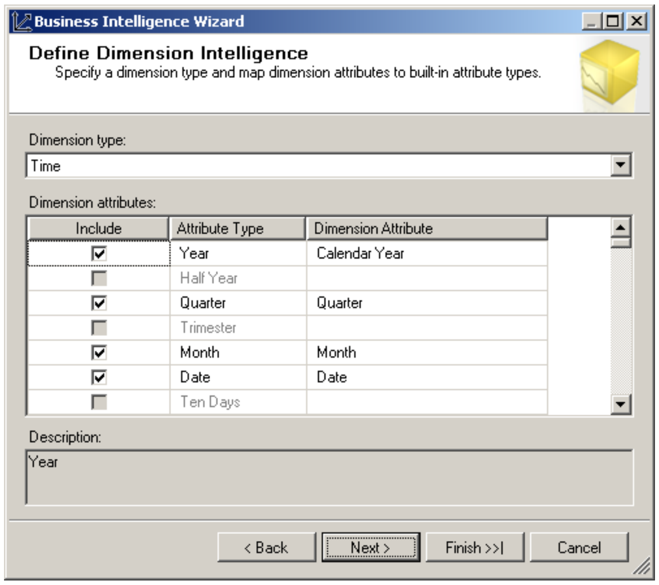
*Figure 33: Parent Member Data Missing from Query Results*

## Attribute Types

In the Dimension Wizard, you have the option to assign an Attribute Type to each attribute. In most cases, the default value of **Regular** is sufficient. However, to support specific business scenarios, you can assign a different attribute type.

The most common reason to change attribute types from the default is to identify date-related attributes. Some client tools can perform calculations based on dates when date attributes are identifiable. In addition, certain MDX functions such as QTD() or YTD() require attributes to have attribute types of Quarters or Years, respectively. You might also change attribute types if you have an Account dimension and need to identify accounts as assets, liabilities, revenue, and expenses to instruct Analysis Services how to properly aggregate values by account type. Some client tools recognize geographical attribute types such as city or country to support the display of cube values in a map.

You can change the **Attribute Type** value in the **Dimension Wizard** or by setting the **Type** property in the **Properties** window for each attribute. Either of these methods can be a bit tedious, so a third option is to use the **Business Intelligence Wizard**, which you launch by clicking the first button in the **Dimension Designer** toolbar. You then select the **Define Dimension Intelligence** option on the **Choose Enhancement** page of the wizard and the **Dimension Type** on the **Define Dimension Intelligence** page. On this latter page, you select the check box for each type of attribute that exists in the dimension and map a standard attribute type to your dimension's attribute, as shown in Figure 34. When you close the wizard, the **Type** property of each attribute you mapped is updated to reflect the new assignment.



Define Dimension Intelligence

Specify a dimension type and map dimension attributes to built-in attribute types.

Dimension type: Time

Dimension attributes:

| Include | Attribute Type | Dimension Attribute |
|---|---|---|
| ✓ | Year | Calendar Year |
|  | Half Year |  |
| ✓ | Quarter | Quarter |
|  | Trimester |  |
| ✓ | Month | Month |
| ✓ | Date | Date |
|  | Ten Days |  |

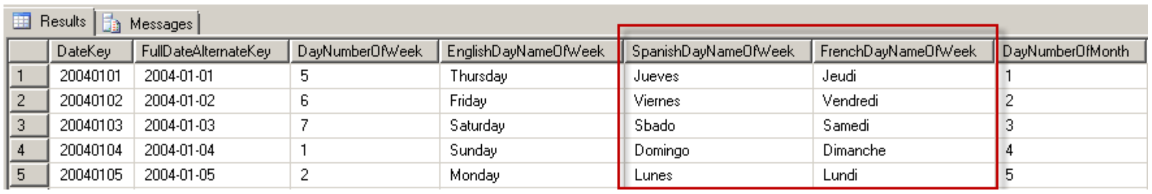
Description:

Year

*Figure 34: Attribute Type Mapping*

## Translations

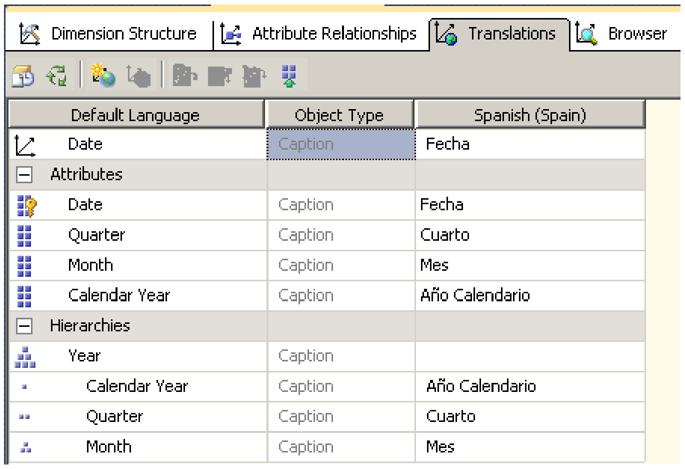
If you are required to support multiple languages to accommodate a variety of users, you can configure translations for the dimension. Although this feature works transparently if the front-end tool supports it, Analysis Services does not automatically perform translations. You must include the translation for each member name in the dimension's relational source. The source must contain one column for each language and each attribute that you want to translate, as shown in Figure 35.



| | DateKey | FullDateAlternateKey | DayNumberOfWeek | EnglishDayNameOfWeek | SpanishDayNameOfWeek | FrenchDayNameOfWeek | DayNumberOfMonth |
|---|---|---|---|---|---|---|---|
| 1 | 20040101 | 2004-01-01 | 5 | Thursday | Jueves | Jeudi | 1 |
| 2 | 20040102 | 2004-01-02 | 6 | Friday | Viernes | Vendredi | 2 |
| 3 | 20040103 | 2004-01-03 | 7 | Saturday | Sbado | Samedi | 3 |
| 4 | 20040104 | 2004-01-04 | 1 | Sunday | Domingo | Dimanche | 4 |
| 5 | 20040105 | 2004-01-05 | 2 | Monday | Lunes | Lundi | 5 |

*Figure 35: Translation Columns in a Dimension Table*

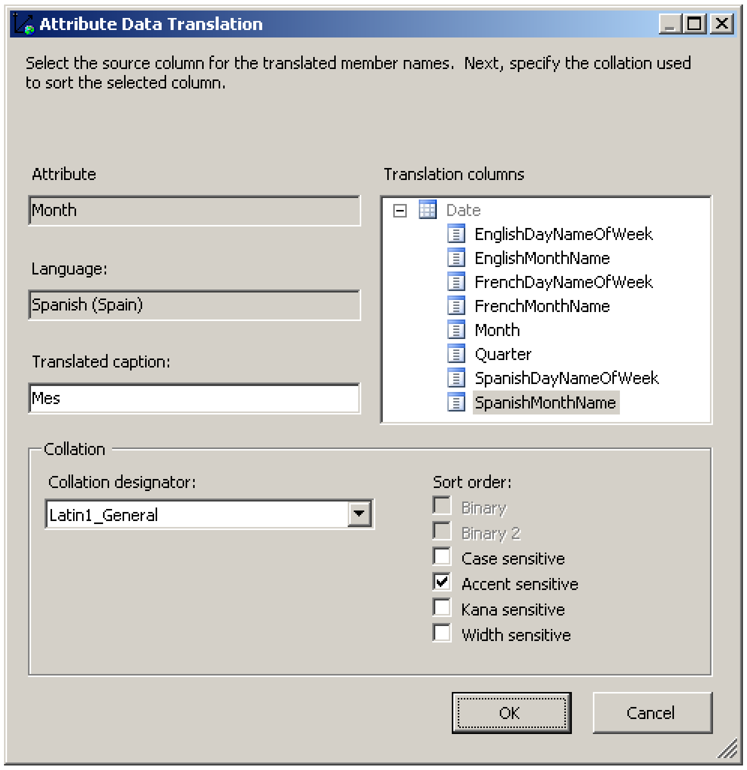
Then you have two steps to perform to configure a dimension to use translations. First, open the **Translations** tab of the Dimension Designer and click the third button in the toolbar labeled **New Translation**. You must then type the caption to display for each attribute, as shown in Figure 36.



| Default Language | Object Type | Spanish (Spain) |
|---|---|---|
| Date | Caption | Fecha |
| Attributes | | |
| Date | Caption | Fecha |
| Quarter | Caption | Cuarto |
| Month | Caption | Mes |
| Calendar Year | Caption | Año Calendario |
| Hierarchies | | |
| Year | Caption | |
| Calendar Year | Caption | Año Calendario |
| Quarter | Caption | Cuarto |
| Month | Caption | Mes |

*Figure 36: Translation Captions*

Second, click the cell at the intersection of the translation and the dimension object, such as the cell in the Attributes list containing the caption "*Mes*", and then click the ellipsis button to open the **Attribute Data Translation** dialog box for the attribute. In this dialog box, you map the attribute to the column containing the applicable translation, as shown in Figure 37. In this way, you can add as many languages as you need to support in your Analysis Services database.



Attribute Data Translation

Select the source column for the translated member names. Next, specify the collation used to sort the selected column.

Attribute: Month

Language: Spanish (Spain)

Translated caption: Mes

Translation columns:
- Date
  - EnglishDayNameOfWeek
  - EnglishMonthName
  - FrenchDayNameOfWeek
  - FrenchMonthName
  - Month
  - Quarter
  - SpanishDayNameOfWeek
  - SpanishMonthName

Collation

Collation designator: Latin1_General

Sort order:
- [ ] Binary
- [ ] Binary 2
- [ ] Case sensitive
- [✓] Accent sensitive
- [ ] Kana sensitive
- [ ] Width sensitive

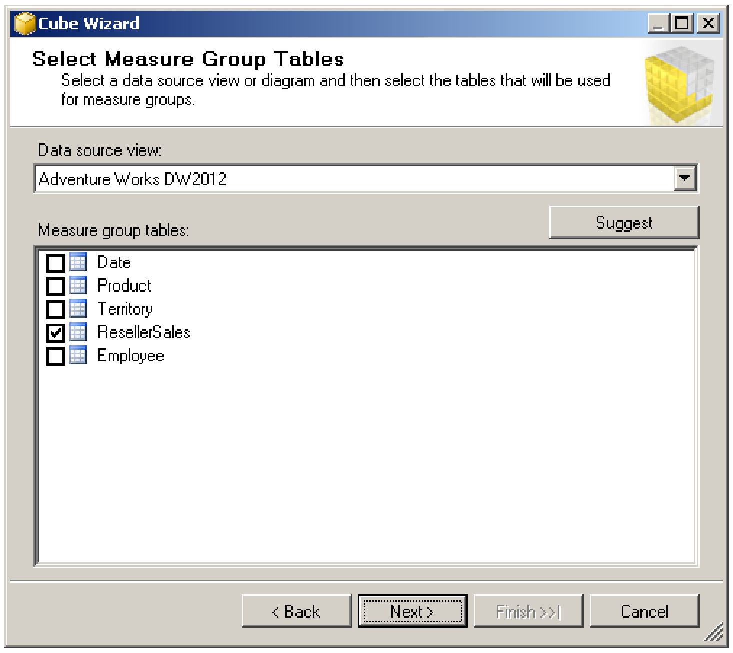
*Figure 37: Translation Column Mapping*

# Chapter 4  Developing Cubes

Generally, I prefer to build most, if not all, dimensions prior to developing cubes, but it's also possible to generate skeleton dimension objects when you use the Cube Wizard. As I explain in this chapter, you use the wizard to select the tables that become measure groups in the cube and identify the dimensions to associate with the cube. After completing the wizard, you configure properties for the measures and adjust the relationships between dimensions and measure groups, if necessary. As part of the cube development process, you can also configure partitions and aggregations to manage storage and query performance. Optionally, you can add perspectives to provide alternate views of the cube and translations to support multiple languages.

## Cube Wizard

The Cube Wizard provides a series of pages to jumpstart the development of a cube. To start the Cube Wizard, right-click the **Cubes** folder in Solution Explorer. On the **Select Creation Method** page of the wizard, keep the default selection, **Use Existing Tables**. Just like the Dimension Wizard, the Cube Wizard offers other options to use when you want to generate tables to match specified requirements. On the **Select Measure Group Tables** page, shown in Figure 38, all tables in the DSV are visible and you choose those you want to use as a measure group table. Normally, a measure group table is the same as a fact table in a star schema, but that term is not used here because you can use data from other structures. In general, a measure group table is a table containing numeric values that users want to analyze and also has foreign key relationships to the dimension tables in the DSV.


Cube Wizard

Select Measure Group Tables

Select a data source view or diagram and then select the tables that will be used for measure groups.

Data source view: Adventure Works DW2012

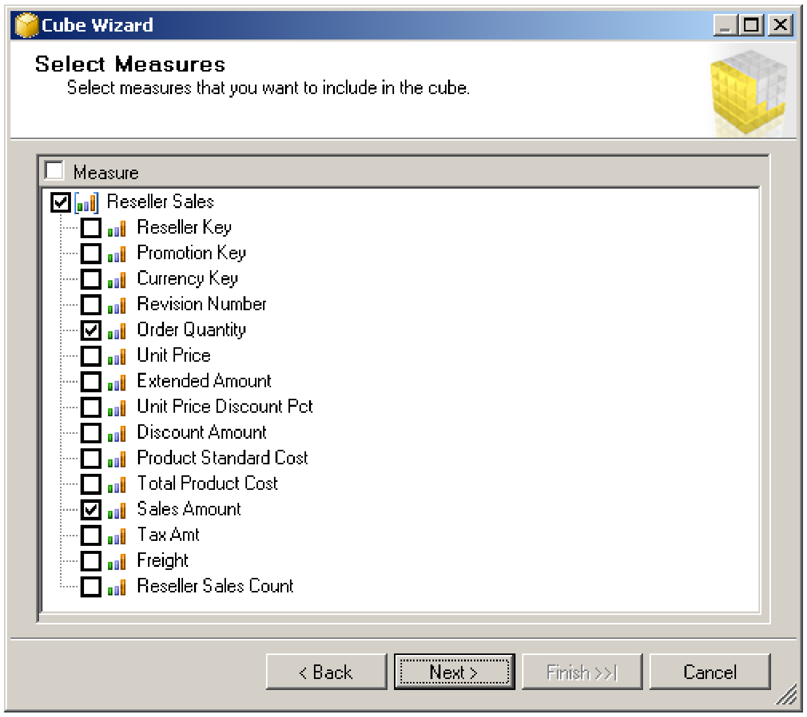
Measure group tables:

- [ ] Date
- [ ] Product
- [ ] Territory
- [x] ResellerSales
- [ ] Employee

*Figure 38:  Select Measure Group Tables Page of Cube Wizard*

***Tip: If you're not sure which tables to select, click the Suggest button. The wizard examines the relationships between tables in the DSV, and proposes tables that meet the general characteristics of a fact table. This feature is useful for people who are not professional business intelligence developers but want to build a cube.***

After selecting the measure group table, you see a list of all the columns in each table that have a numeric data type on the next page of the wizard, as shown in Figure 39. Some columns might not be measures, but are instead keys to dimensions, so be sure to review the list before selecting all items. Furthermore, you are not required to add all measures in a table to your cube.


Cube Wizard

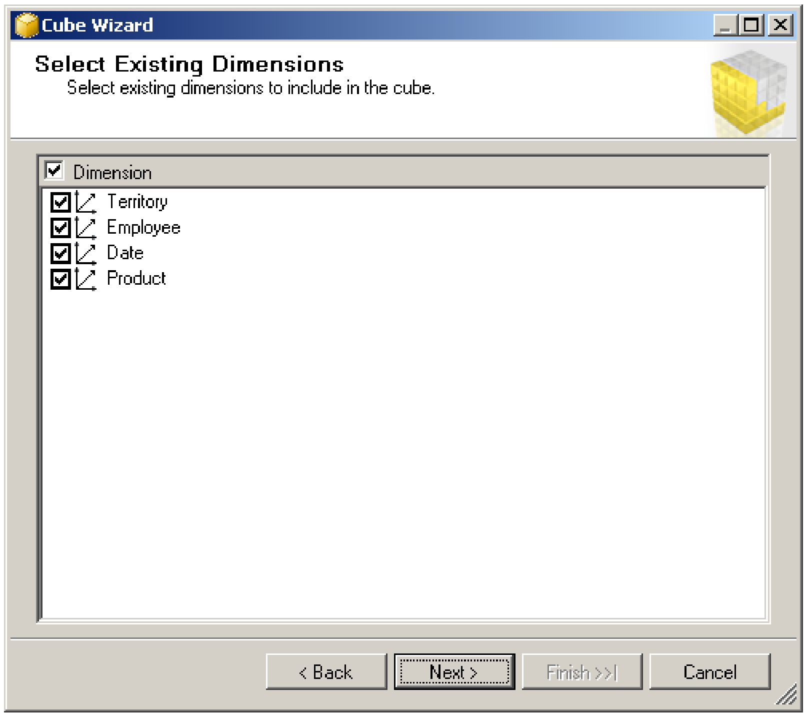
Select Measures

Select measures that you want to include in the cube.

- [ ] Measure
- [✓] Reseller Sales
  - [ ] Reseller Key
  - [ ] Promotion Key
  - [ ] Currency Key
  - [ ] Revision Number
  - [✓] Order Quantity
  - [ ] Unit Price
  - [ ] Extended Amount
  - [ ] Unit Price Discount Pct
  - [ ] Discount Amount
  - [ ] Product Standard Cost
  - [ ] Total Product Cost
  - [✓] Sales Amount
  - [ ] Tax Amt
  - [ ] Freight
  - [ ] Reseller Sales Count

*Figure 39: Select Measures Page of Cube Wizard*

The next page of the wizard displays dimensions that already exist in the database, as shown in Figure 40. That is, the list includes the dimensions currently visible in the Solution Explorer. You can select some or all dimensions to add to the cube.


Cube Wizard
Select Existing Dimensions
Select existing dimensions to include in the cube.
< Back
Next >
Finish >>|
Cancel

☑ Dimension
☑ Territory
☑ Employee
☑ Date
☑ Product

*Figure 40: Select Existing Dimensions Page of Cube Wizard*

***Note: If the wizard detects that a dimension could be created from the measure group table, you will see another page in the wizard before it finishes. That's an advanced design for many-to-many relationships that is not covered in this book. If you are not familiar with handling many-to-many relationships, clear the selections on this page. To learn more about many-to-many relationships, see*** The Many-to-Many Revolution 2.0***.***

After you finish the Cube Wizard, you see the results in the Cube Designer, shown in Figure 41. The Cube Designer has many different tabs, several of which will be explored in this chapter, and others that will be explored in Chapter 5, "Enhancing cubes with MDX," and Chapter 6, "Managing Analysis Services databases." The first tab is for working with the cube structure. In the upper left corner, you see all the measures that you added to the cube thus far. By default, these measures appear in a tree view. If you have multiple measure groups in the cube, then you see each set of measures organized below its respective measure group.

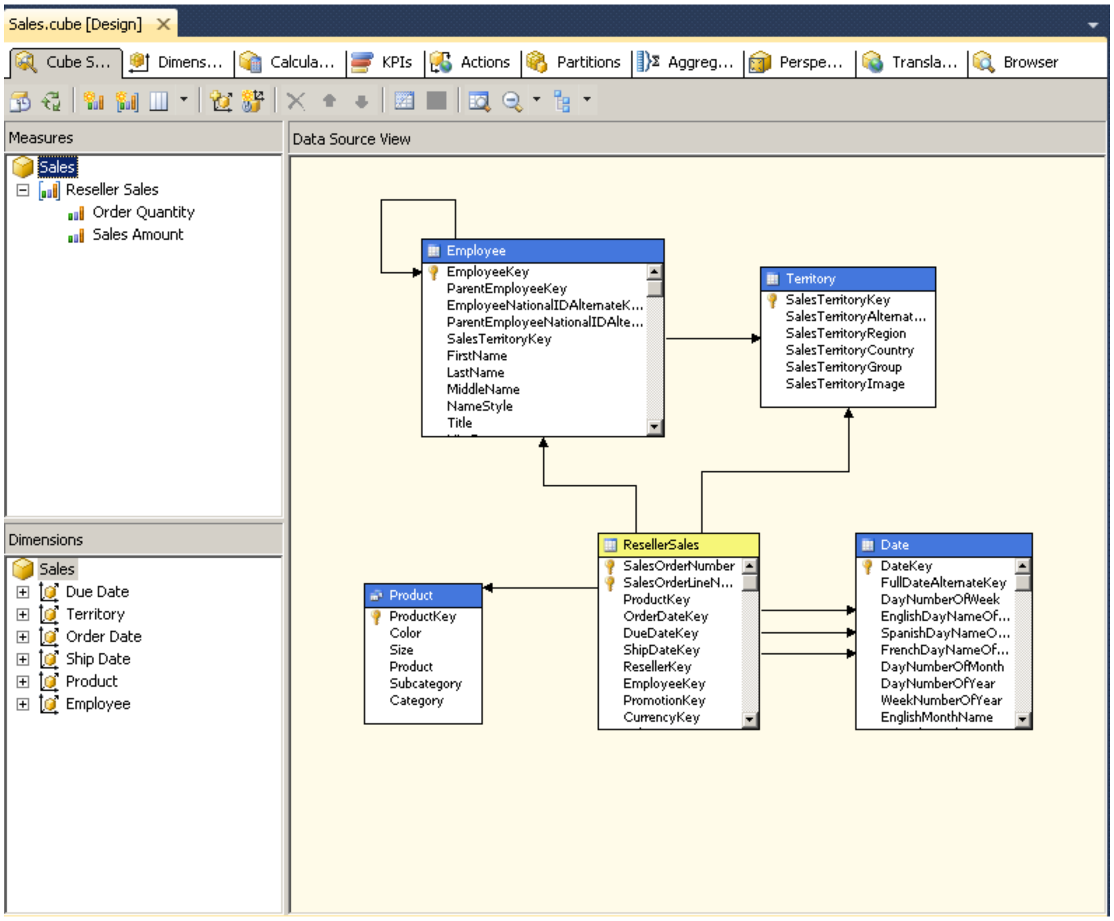


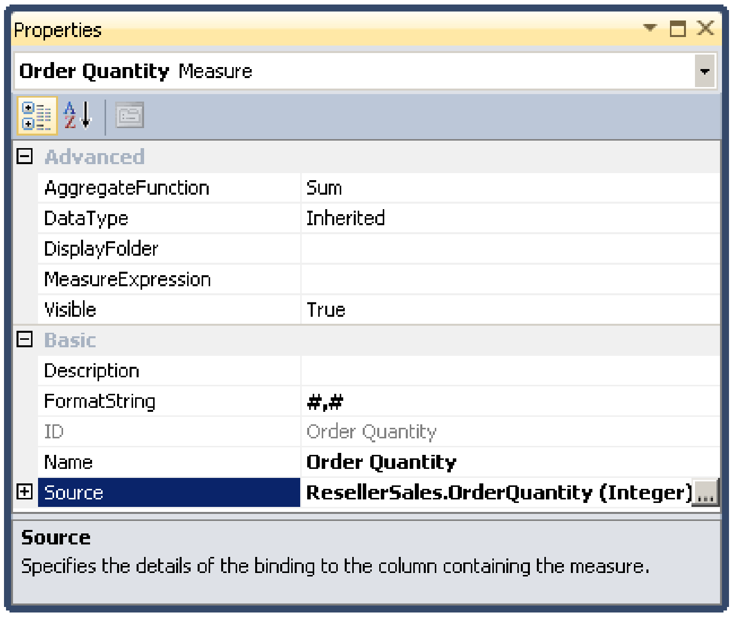
*Figure 41: Cube Structure Tab of Cube Designer*

## Measures

The first task to perform after building a cube is to review the properties for each measure and adjust the settings where appropriate. You should especially take care to select the appropriate aggregate function. If you later decide to add more measures to the cube, you can easily add a single measure or add multiple measures by adding a new measure group.

### Measure Properties

At minimum, you should review the following properties, as shown in Figure 42: **Name**, **FormatString**, **Visible**, **DisplayFolder**, and **AggregateFunction**.


Properties

**Order Quantity** Measure

| | |
|---|---|
| **Advanced** | |
| AggregateFunction | Sum |
| DataType | Inherited |
| DisplayFolder | |
| MeasureExpression | |
| Visible | True |
| **Basic** | |
| Description | |
| FormatString | **#,#** |
| ID | Order Quantity |
| Name | **Order Quantity** |
| Source | **ResellerSales.OrderQuantity (Integer)** |

**Source**
Specifies the details of the binding to the column containing the measure.

*Figure 42: Measure Properties*

The first thing to check for each measure is its **Name** property**.** Make sure the name appears correctly and is unambiguous for users. Measure names must be unique across all measure groups in the cube.

Next, set the **FormatString** property for each measure to make it easier to read the values when you test the cube in a cube browser such as Excel. You can assign a built-in format string such as **Currency** or **Percent**, or you can use a custom formatting string just as you can in Excel.

You can also choose to hide a measure by changing its **Visible** property to **False**. If you do this, the measure continues to be available for use in calculations, but anyone browsing the cube cannot see the measure.

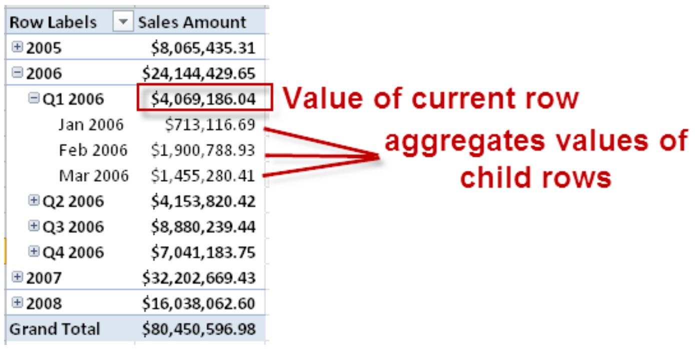
If a measure group contains a long list of measures, you can organize them into groups by specifying a **DisplayFolder** value. The first time you reference a folder, you have to type in the name directly. Afterwards, the folder is available in a drop-down list so that you can assign it to other measures.

Last, review the most important property of all, **AggregateFunction**. The default is **Sum**, which instructs Analysis Services to add up each record in the fact table by dimension when returning results for a query. In the next section, I explain each of the available aggregate functions.

## Aggregate Functions

There are several different types of aggregate functions available:

- **Sum**. The most commonly used aggregate function is the **Sum** function. When you use the function in a query, a list of dimension members is displayed in rows or columns (or both) and the measure is displayed at the intersection. When a dimension member is either the All level or a level in a user hierarchy that has children, the value of the measure is the sum of the children's values for the member that you see on a particular row, as shown in Figure 43. In other words, the value for Q1 2006 is the sum of the values for Jan 2006, Feb 2006, and Mar 2006. Likewise, the value for 2006 is the sum of the values for Q1 2006, Q2 2006, Q3 3006, and Q4 2006.



| Row Labels | Sales Amount |
|---|---|
| ⊞2005 | $8,065,435.31 |
| ⊟2006 | $24,144,429.65 |
| ⊟Q1 2006 | $4,069,186.04 |
| Jan 2006 | $713,116.69 |
| Feb 2006 | $1,900,788.93 |
| Mar 2006 | $1,455,280.41 |
| ⊞Q2 2006 | $4,153,820.42 |
| ⊞Q3 2006 | $8,880,239.44 |
| ⊞Q4 2006 | $7,041,183.75 |
| ⊞2007 | $32,202,669.43 |
| ⊞2008 | $16,038,062.60 |
| Grand Total | $80,450,596.98 |

*Figure 43: Aggregating Measures by Dimension Members*

- **Count**. This is another common function. You use it to get a count of the non-empty rows in the fact table associated with the children of the current dimension member.

- **Minimum, Maximum**. You use these functions to find the minimum or maximum values of a member's children, although these are less commonly used functions.

- **DistinctCount**. This function returns the number of unique members in a dimension that are found in a fact table. When you assign a distinct count function to a measure, it gets placed in a separate measure group to optimize performance. For the best performance, use only integer values for distinct count.

- **None**. When you have measures that are calculations that cannot be decomposed for some reason, such as when you have a percentage value that you want to include in a cube for use in other calculations, you can prevent aggregation for that measure by setting the **Aggregate** function to **None**.

There are several aggregate functions that are considered semi-additive. When you use a semi-additive aggregate function, the measure aggregates by using the **Sum** function as long as the current dimension is not the **Date** dimension. Within the **Date** dimension, the aggregation behavior depends on which semi-additive function you assigned to the measure:

- **ByAccount**. When you use this aggregate function, your cube must also include a dimension with the Account type and set the attribute types correctly to correctly identify accounts as Assets, Liabilities, Revenue, and Expenses, for example. With this aggregate function, the fact table values are added up properly by time according to the type of account that is displayed. For more information, see Add Account Intelligence to a Dimension.

- **AverageOfChildren**. As an example, if the current dimension member is a member of the quarter level of a calendar year hierarchy such as Q2 2008, Analysis Services uses this aggregate function to sum up its three months' values (April 2008, May 2008, and June 2008), and divide the result by 3.

- **FirstChild, LastChild**. These aggregate functions are useful for a snapshot measure, such as an inventory count for which you load the fact table with a balance on the first day or last day of a month. Then, Analysis Services responds to a query by month with the value on the first or last day, whereas it returns the value for the first or last month for a query by quarter.

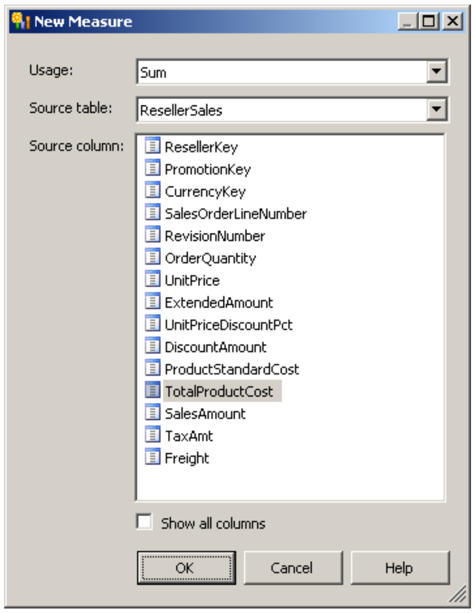
- **FirstNonEmpty, LastNonEmpty**. These functions behave much like **FirstChild** and **LastChild**. However, rather than using the first or last child explicitly, Analysis Services uses the first or last child only if it has a value. If it doesn't, Analysis Services continues looking at each subsequent (or previous) child until it finds one that is not empty. For example, with **FirstNonEmpty**, Analysis Services looks at the second child, third child, and so on until it finds a child with a value. Likewise, with **LastNonEmpty**, it looks at the next to last child, the one before that, and the one before that, and so on.

***Note: It's important to know that the semi-additive functions are available only in Enterprise and Business Intelligence editions for production. You can also use Developer's Edition to create a project that you later deploy to a production server. An exception is the LastChild function, which is available in all editions.***

## Additional Measures

As you develop a cube, you might find that you need to create additional measures for your cube. As long as the measures are in the DSV, you can right-click anywhere in the **Measures** pane on the **Cube Structure** tab of the Cube Designer, and select one of the following options:

- **New Measure**. Use this option to add a single measure from the dialog box, as shown in Figure 44. You must also specify usage, which sets the aggregate function.



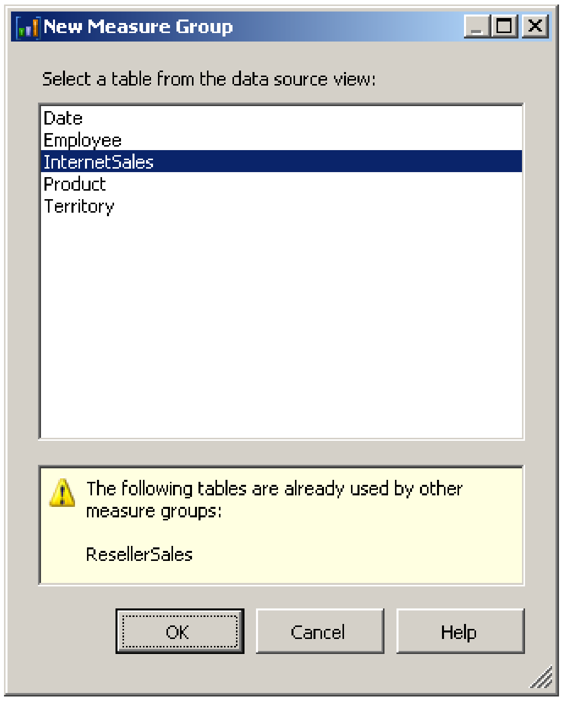
Usage: Sum

Source table: ResellerSales

Source column:
ResellerKey
PromotionKey
CurrencyKey
SalesOrderLineNumber
RevisionNumber
OrderQuantity
UnitPrice
ExtendedAmount
UnitPriceDiscountPct
DiscountAmount
ProductStandardCost
TotalProductCost
SalesAmount
TaxAmt
Freight

*Figure 44: Addition of New Measure*

- **New Measure Group**. With this option, you see a list of the tables currently in the DSV. When you select a table, all the columns with numeric data types are added to the cube as measures. In that case, you might need to delete several columns because it is likely that one or more dimension keys are added as part of the group.



The following tables are already used by other measure groups:

ResellerSales

*Figure 45: Addition of New Measure Group*

## Role-playing Dimension

A special situation arises when you have a dimension used multiple times within a cube for different purposes, known as a role-playing dimension. Before I show you what a role-playing dimension looks like, let me explain the difference between database and cube dimensions and the data structure necessary in the DSV to support a role-playing dimension in a cube.

Consider the AdventureWorksDW2012 sample database, which contains only a single date table, DimDate, for which you can create a dimension object in an Analysis Services project. Any dimension that appears in Solution Explorer is a database dimension. Not only can you associate a database dimension with one or more cubes that you add to the database, but you can also configure different properties for each instance of the dimension within a cube. However, each cube dimension continues to share common properties of the dimension at the database level.

To structure data to support a role-playing dimension, the fact table in the DSV has multiple columns with a relationship to the same dimension table, as shown in Figure 46. In this example, the Reseller Sales table (a fact table) has multiple relationships with the DateKey column in the Date table, based on the fact table's OrderDateKey, DueDateKey, and ShipDateKey. Each column has a different meaning. The OrderDateKey stores the value for the date of sale, the DueDateKey stores the value for the promised delivery date, and the ShipDateKey is the date of shipment. Each column relates to the same date dimension because any given date has the same properties—the same month, the same quarter, the same year, day of year, and so on.

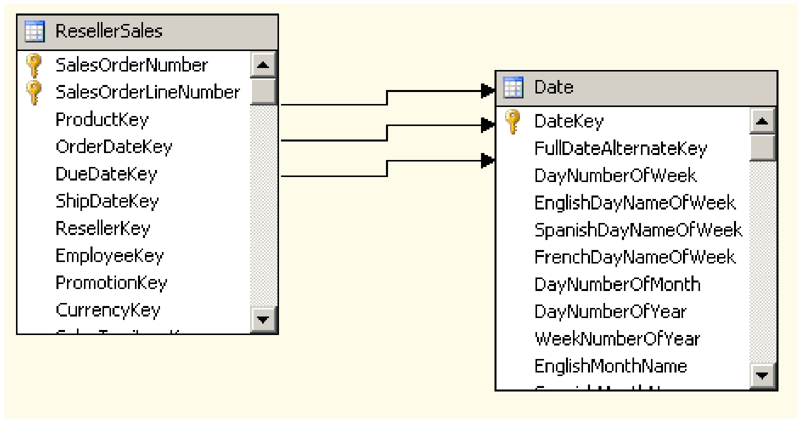


*Figure 46: Multiple Foreign Key Relationships for Role-Playing Dimensions*

When you add the Date dimension to the cube, the presence of the multiple relationships triggers the addition of multiple versions of the dimension to the cube as role-playing dimensions, as shown in Figure 47. You can rename each role-playing dimension if you like, or hide or delete them as determined by the users' analysis requirements. You rename the dimension on the **Cube Structure** tab, because the name of a cube dimension, whether or not it's a role-playing dimension, is a property set at the cube level and does not affect the dimension's name at the database level.

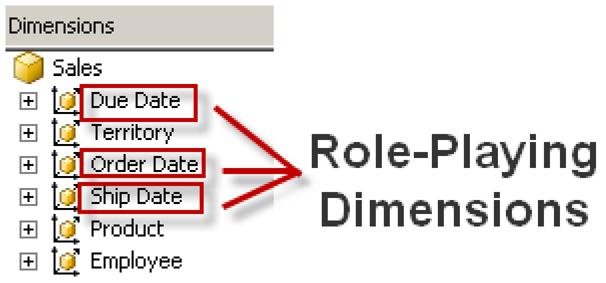
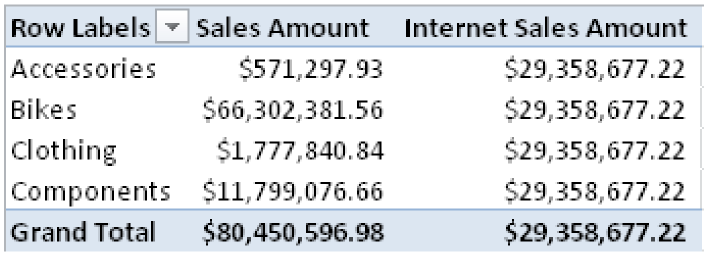


*Figure 47: Role-Playing Dimensions in Analysis Services*

## Dimension Usage

Much like you define foreign key relationships between fact and dimension tables in the DSV, you also define relationships between measure groups and cube dimensions on the **Dimension Usage** tab of the Cube Designer. In fact, if you have foreign key relationships correctly defined in the DSV, the dimension usage relationships are usually correctly defined without further intervention on your part. However, sometimes a relationship is not automatically detected, more commonly when you add measure groups to an existing cube than when you use the Cube Wizard.

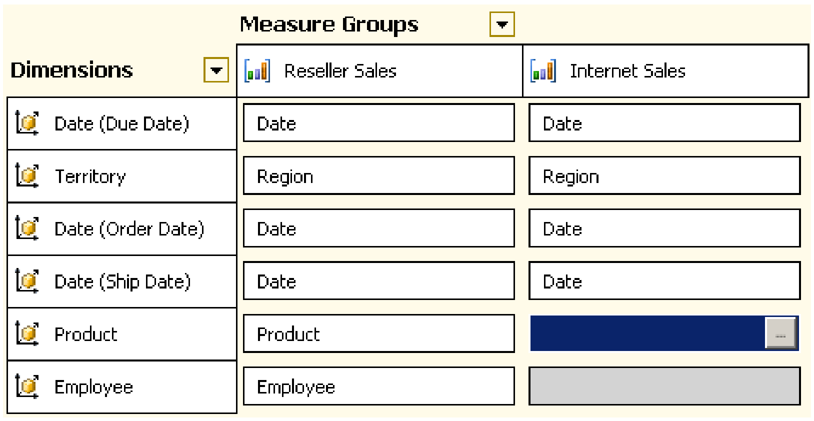
You can recognize a problem in dimension usage when you see the grand total value for a measure repeat across dimension members when you expect to see separate values. For example, in Figure 48, the Sales Amount column displays separate values for each category, whereas the Internet Sales Amount column repeats the grand total value.



| Row Labels | Sales Amount | Internet Sales Amount |
|---|---|---|
| Accessories | $571,297.93 | $29,358,677.22 |
| Bikes | $66,302,381.56 | $29,358,677.22 |
| Clothing | $1,777,840.84 | $29,358,677.22 |
| Components | $11,799,076.66 | $29,358,677.22 |
| **Grand Total** | **$80,450,596.98** | **$29,358,677.22** |

*Figure 48: Aggregation Error Due to Missing Relationship*

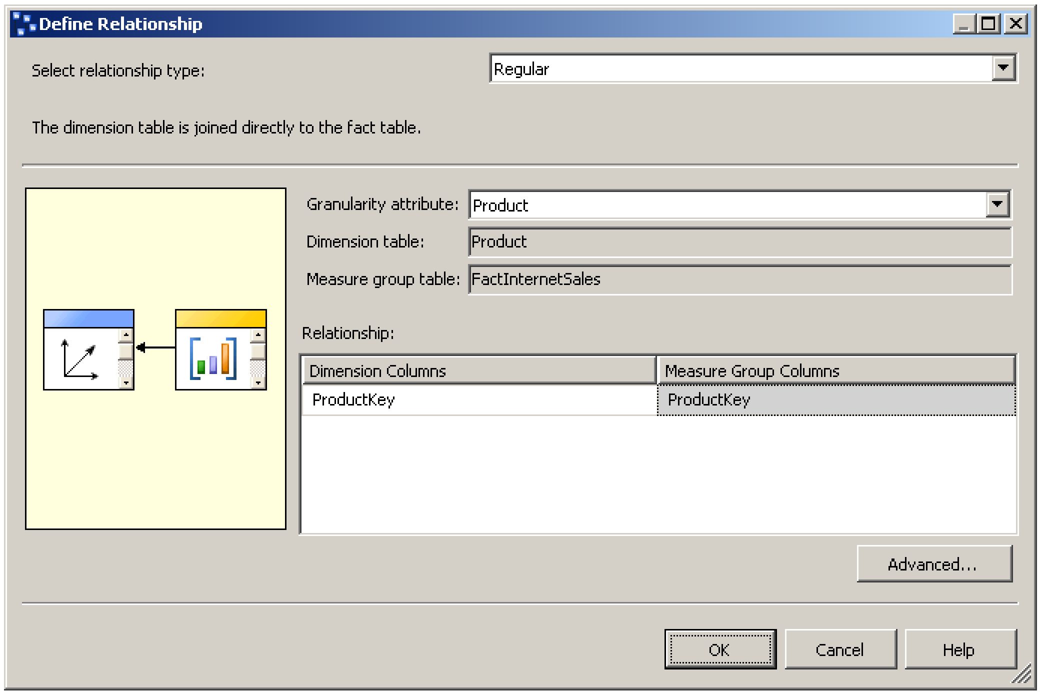
On the Dimension Usage tab of the Cube Designer, as shown in Figure 49, you can see that the intersection between the Product dimension and the Internet Sales measure group is empty. Dimension usage is the link between the measures in the measure group and the dimension. If the link is missing, Analysis Services is unable to aggregate fact table records by dimension member, and can only display the aggregate value for all fact table records as a whole.



| | Measure Groups | |
|---|---|---|
| Dimensions | Reseller Sales | Internet Sales |
| Date (Due Date) | Date | Date |
| Territory | Region | Region |
| Date (Order Date) | Date | Date |
| Date (Ship Date) | Date | Date |
| Product | Product | |
| Employee | Employee | |

*Figure 49: Dimension Usage Tab of Cube Designer*

To add the correct relationship, click the intersection between a dimension and a measure group, and then click the ellipsis button that is displayed. In the **Define Relationship** dialog box, shown in Figure 50, select **Regular** in the **Select Relationship Type** drop-down list, and then select the attribute that has a key column defined as a foreign key column in the fact table. Last, select that foreign key in the **Measure Group Column** drop-down list.


Define Relationship

Select relationship type: Regular

The dimension table is joined directly to the fact table.

Granularity attribute: Product

Dimension table: Product

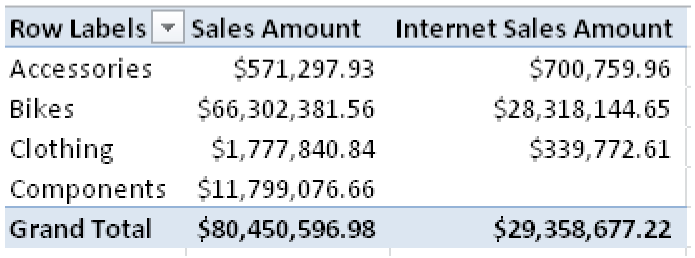
Measure group table: FactInternetSales

Relationship:

| Dimension Columns | Measure Group Columns |
|---|---|
| ProductKey | ProductKey |

*Figure 50: Dimension Usage Relationship Definition*

After you deploy the project to update the cube, you can test the results by browsing the cube. With the relationship defined for dimension usage, the values for Internet Sales Amount are now displayed correctly, as shown in Figure 51.



| Row Labels | Sales Amount | Internet Sales Amount |
|---|---|---|
| Accessories | $571,297.93 | $700,759.96 |
| Bikes | $66,302,381.56 | $28,318,144.65 |
| Clothing | $1,777,840.84 | $339,772.61 |
| Components | $11,799,076.66 | |
| Grand Total | $80,450,596.98 | $29,358,677.22 |

*Figure 51: Corrected Aggregation*

## Partitions

Analysis Services stores fact data in an object called a partition. You can use multiple partitions as a way to optimize the query experience for users and administrative tasks on the server, but you need to understand how to develop a proper partitioning strategy to achieve your goals. The main reason to partition is to manage physical storage, so it's important to understand the different options that you have. Then you can perform the steps necessary to design a partition and to combine partitions if you later need to change the physical structure of your cube.

## Partitioning Strategy

To the end user, a cube looks like a single object, but you can manage the data in a cube as separate partitions. More accurately, each measure group is in a separate partition, but you can create multiple partitions to achieve specific performance and management goals for a cube. There are three main reasons why you should consider dividing a measure group into separate partitions:

- **Optimize storage**. You can define different storage modes for each partition: MOLAP, HOLAP, or ROLAP. Your choice affects how and whether data is stored on disk. You can also place each physical partition on a different drive or even on different servers to distribute disk I/O operations. I explain more about MOLAP, HOLAP, and ROLAP storage modes in the next section of this chapter.

- **Optimize query performance**. You can also design partitions to control the contents of each partition. When a query can find all data in a single partition, it retrieves that data more quickly. But even if the data is spread across multiple partitions, the partition scans can occur in parallel and occur much faster than a scan of a single large partition. In addition, you can set different aggregation levels for each partition, which can affect the amount of extra storage required to store aggregations and also how much faster queries can run.

- **Optimize processing**. Often the data that you load into a cube does not change. That means you can load it once and leave it there. Processing is the mechanism that loads data into the cube, which you will learn more about in Chapter 6. If you separate data into partitions, you can restrict processing to a single small partition to load current data and leave other unchanged data in place in separate partitions. That makes processing execute more quickly. Furthermore, even if you have to process multiple partitions, you can process them in parallel and thereby reduce the overall time required to process. The amount of parallelism that you can achieve depends on the server hardware that you use for Analysis Services.

When you create a partitioning strategy, you can use different storage modes, aggregation levels, and assign a variable number of rows per partition. For example, you can have several months' worth of data in the current year partition, twelve months in a prior year, and 5 years or 60 months in a history partition. Another benefit of this approach is the ability to maintain a moving window of data. You can easily drop off older data and add in new data without disrupting other data that you want to keep in the cube.

## Storage Modes

The default storage mode is called **MOLAP**, which is the abbreviation for Multidimensional OLAP. With MOLAP, as shown in Figure 52, all data, metadata, and aggregations are stored and managed by Analysis Services. When you process the cube, data is read from the relational source according to the rules defined in the UDM, which stands for Unified Dimensional Model. The UDM is metadata that contains all the dimension and cube definitions, describes how the data needs to be stored by Analysis Services, and contains the aggregation design, which the server processes after loading fact data to include aggregations in the data stored on the server. When a client application sends an MDX query to the server, the server responds by querying the data or aggregations in the cube.

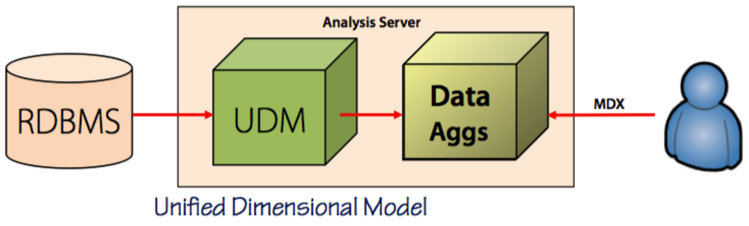


*Figure 52: MOLAP Storage*

***Note: MOLAP storage is an efficient way to store data in a partition. It's highly compressed and indexed, and does not require as much space to store data as the source data.***

Another storage mode is called **HOLAP**, which stands for Hybrid OLAP. With HOLAP, only the aggregation data and the UDM are stored on the server, but data remains in your relational database, as shown in Figure 53. When you process the cube, Analysis Services reads from the relational source and calculates the aggregations, which it then stores on the server. When a client application sends an MDX query to the server, the server responds with a result derived from aggregations if possible, which is the fastest type of query. However, if no aggregation exists to answer the query, the server uses the UDM to translate the MDX query into an SQL query, which in turn is sent to the relational source. In that case, query performance can be slower than it would be if the data were stored in MOLAP.

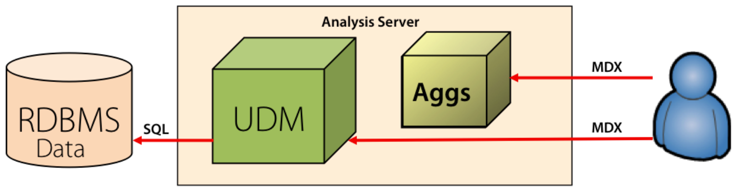


*Figure 53: HOLAP Storage*

HOLAP storage is most often used when a large relational warehouse is in place for other reasons and the majority of queries can be answered by aggregations. If the difference in performance between MOLAP and HOLAP is minimal, HOLAP might be a preferable option when you want to avoid the overhead of additional processing to create a MOLAP store of the data and the number of queries requiring the detail data is small.

The third storage mode is called **ROLAP**, which stands for Relational OLAP. With ROLAP, only the UDM is stored on the server while data and aggregations stay in your relational database, as shown in Figure 54. When you process the cube, no data is read from the relational source, but instead Analysis Services checks the consistency of the partition metadata during processing. Furthermore, if you design aggregations for a ROLAP partition, Analysis Services creates indexed views. As a result, there is no additional copy of data stored in Analysis Services in detail or aggregate form like there is for MOLAP or HOLAP modes.

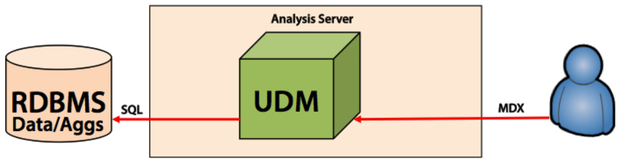


*Figure 54:  ROLAP Storage*

In ROLAP mode, when a client application sends an MDX query to the server, the server translates the request into SQL and returns the results from the source. Query performance is slower than it would be if the data were stored in MOLAP mode. However, the benefit of this approach is access to fresh data for real-time analysis. In MOLAP mode, the data is only as current as the last time the partition was processed.

You can also store dimension data in ROLAP storage mode. When you do this, the dimension data does not get copied to Analysis Services during processing. Instead, dimension data is retrieved only in response to a query. Analysis Services maintains a cache of dimension data that updates with each request. That cache is the first place Analysis Services looks before it issues the SQL query to the relational source. You use this option only for very large dimensions with hundreds of millions of members, which makes it a rarely used feature of Analysis Services.

Your choice of correct storage mode depends on your requirements. Larger data volumes with near-real-time data access requirements benefit from ROLAP mode. In that case, queries to your cube return data as soon as it is inserted into the relational database without the overhead of time required to process the data. On the other hand, MOLAP queries tend to be faster than ROLAP queries.

## Partition Design

When you create a new partition, there are two ways that you can design it: table binding or query binding.

With **table binding**, you have multiple fact tables for a measure group and each fact table contains a separate set of data. For example, you might have a separate fact table for different time periods. In that case, you bind each individual table to a specific Analysis Services partition.

Partitions are not required to have common boundaries. That means one partition could have one month of data, another partition could have one year of data, and a third partition could have five years of data. Regardless, the source fact table needs to contain the data to load into the partition. There is no way to filter out data from a fact table when using the table binding method. The partition contains all the data associated with an entire table.

***Tip: The problem with table binding is that changes to your fact table structure require you to manually update each partition definition to correspond. One way to insulate yourself from this problem is to bind the partitions to views instead of tables.***

The other option is to use **query binding**. That way, you can leave all data in a single table which might be easier to manage when loading the fact table. The query for each partition includes a WHERE clause to restrict the rows that Analysis Services copies from the table to the partition during processing.

***Tip: The potential problem with query binding is that you have to be very careful that no two partitions overlap. Make sure that each WHERE clause gets a unique set of rows. Analysis Services does not check for duplicate rows across partitions and if you inadvertently create this error, the data in the cube will be wrong. You should also be aware that query binding breaks the measure group's relationship to the data source view. Consequently, if you make changes to the data source view later, those changes do not update the partition. You'll have to do this manually. Therefore, setting up partitions should be the last step of the cube development process.***

### Partition Merge

You might decide to combine two or more partitions into one at some point. This process is called merging partitions. Let's say that you have a partitioning strategy that assigns each year to a separate partition. As query patterns change, you might prefer to combine the oldest two partitions into a single partition. To do this, you can use a dialog box in SSMS to merge the partitions manually. If this is a process that you run very infrequently, this is a perfectly fine way to do this task.

If you prefer to automate the process, you can create an XMLA script in Management Studio to run the merge. You can then incorporate the script into an Integration Services package using an Execute DDL task. You can also create variables for the partition identifiers in the script so that you can reuse it to merge other partitions.

## Aggregations

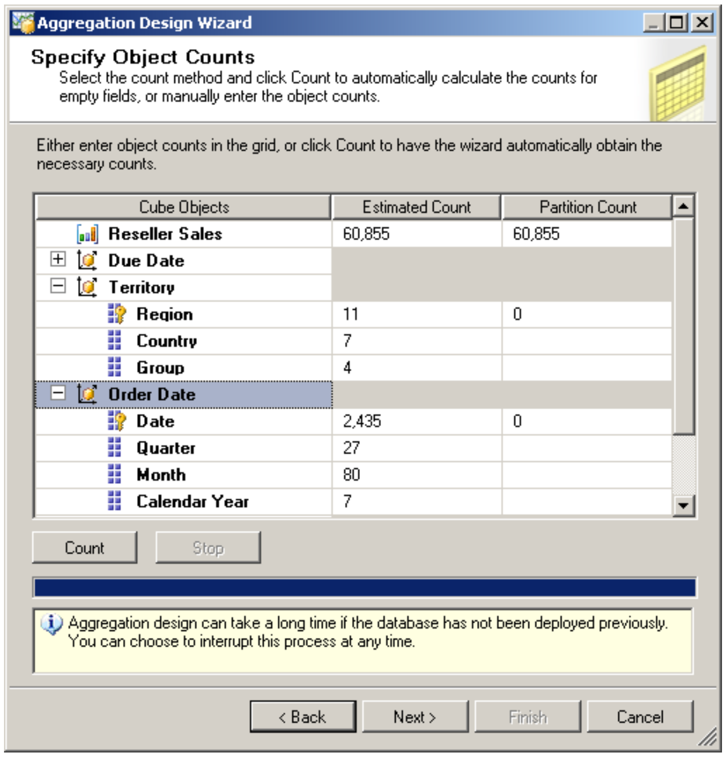
The key to fast query performance in Analysis Services is the addition of aggregations to each cube. Having aggregations available in the Analysis Services database is much like having summary tables in a data warehouse, except you don't have to build or maintain the aggregations as you do with summary tables. Analysis Services does most of the work.

## Aggregation Wizard

You begin the process of adding aggregations by launching the Aggregation Wizard. To do this, open the **Aggregations** tab of the Cube Designer and click the first toolbar button. On the **Review Aggregation Usage** page of the wizard, you can adjust settings to fine-tune the aggregation design process, but in most cases the default settings are adequate.

On the **Specify Object Counts** page of the wizard, shown in Figure 55, you can click **Count** to retrieve actual row counts from the tables associated with the measure group and dimension objects, or you can type an estimated count to more accurately reflect your production environment when you are working against a development database. The cube objects list includes the measure group and attributes included in a user-defined hierarchy.

The Aggregation Wizard uses the estimated count to calculate the ratio of members in an attribute relative to the fact table rows and determine whether the attribute is a candidate for aggregation. For example, because the number of members for the **Group** attribute is 4 and the number of records in **FactResellerSales** is 60,855 (when using the AdventureWorksDW2012 database as a source), there is a high probability (but no guarantee) that the Aggregation Wizard will generate at least one aggregation that includes **Group**. In general, if the number of attribute members is 30 percent or less than the fact table records, Analysis Services considers the attribute as a candidate for aggregation.



Specify Object Counts

Select the count method and click Count to automatically calculate the counts for empty fields, or manually enter the object counts.

Either enter object counts in the grid, or click Count to have the wizard automatically obtain the necessary counts.

| Cube Objects | Estimated Count | Partition Count |
|---|---|---|
| Reseller Sales | 60,855 | 60,855 |
| Due Date | | |
| Territory | | |
| Region | 11 | 0 |
| Country | 7 | |
| Group | 4 | |
| Order Date | | |
| Date | 2,435 | 0 |
| Quarter | 27 | |
| Month | 80 | |
| Calendar Year | 7 | |

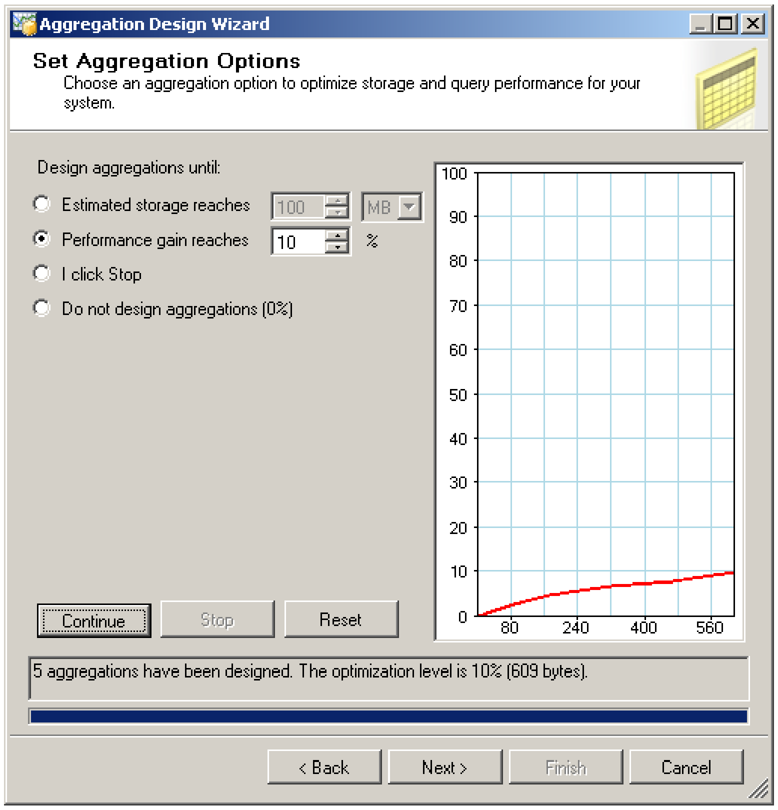
Aggregation design can take a long time if the database has not been deployed previously. You can choose to interrupt this process at any time.

*Figure 55: Specify Object Counts Page of Aggregation Wizard*

On the next page, shown in Figure 56, you select one of the following aggregation options:

- **Estimated Storage**. Use this option to constrain the disk space requirements for aggregations.

- **Performance Gain**. Use this option to add enough aggregations to improve query speed by a specified percentage as compared to a query against a cube having no aggregations. The default is 30 percent, but you should begin with a value between 5 and 10 percent.
- **Stop**. Use this option to interrupt the aggregation design process by clicking the **Stop** button.
- **Do Not Design Aggregations**. Use this option or click **Cancel** to stop the wizard.


Aggregation Design Wizard
Set Aggregation Options
Choose an aggregation option to optimize storage and query performance for your system.
Design aggregations until:
Estimated storage reaches
100
MB
Performance gain reaches
10
%
I click Stop
Do not design aggregations (0%)
Continue
Stop
Reset
100
90
80
70
60
50
40
30
20
10
0
80
240
400
560
< Back
Next >
Finish
Cancel

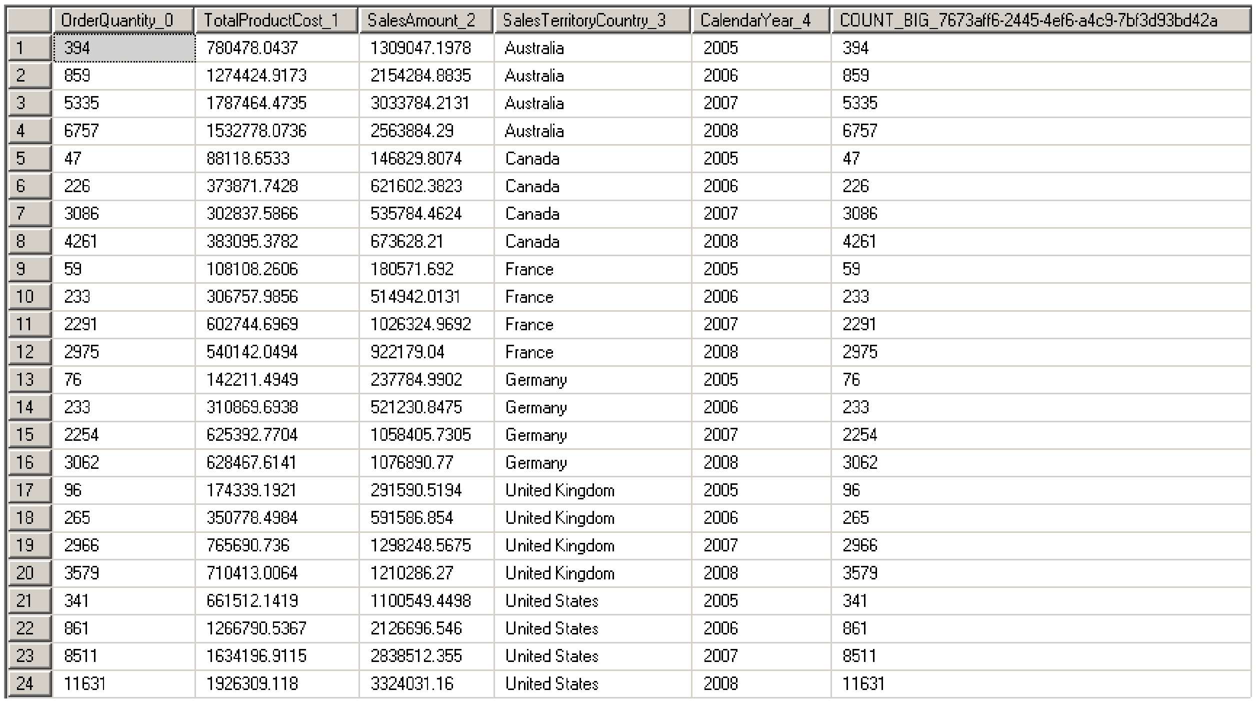
5 aggregations have been designed. The optimization level is 10% (609 bytes).

*Figure 56: Specify Object Counts Page of Aggregation Wizard*

***Note: For more information, see Aggregation Design Strategy. Although the article is written for SQL Server 2005 Analysis Services, the principles remain applicable to all later versions through SQL Server 2012 Analysis Services.***

On the last page of the wizard you have the option to deploy the changes and compute aggregations, or to save the aggregation design without deploying the changes to the server (in case you want to make other modifications to the database design first). By completing the wizard, you add one or more aggregation designs to your project. If you also deploy the aggregations, Analysis Services uses this aggregation design to calculate aggregations and store the results on disk.

An aggregation design is a template that Analysis Services uses to define the dimensions to include with aggregations. Although you can see the aggregation design in SSDT or the database script, you cannot see the aggregation values themselves if you use the default MOLAP storage mode for measure group partitions. However, if you use ROLAP storage, you can see the indexed views that Analysis Services uses for aggregations, such as the one shown in Figure 57. To set up ROLAP storage, click the **Storage Settings** link on the **Partitions** tab of the Cube Designer for the measure group, select **Real-time ROLAP**, click the **Options** button, and select the **Enable ROLAP Aggregations** check box.



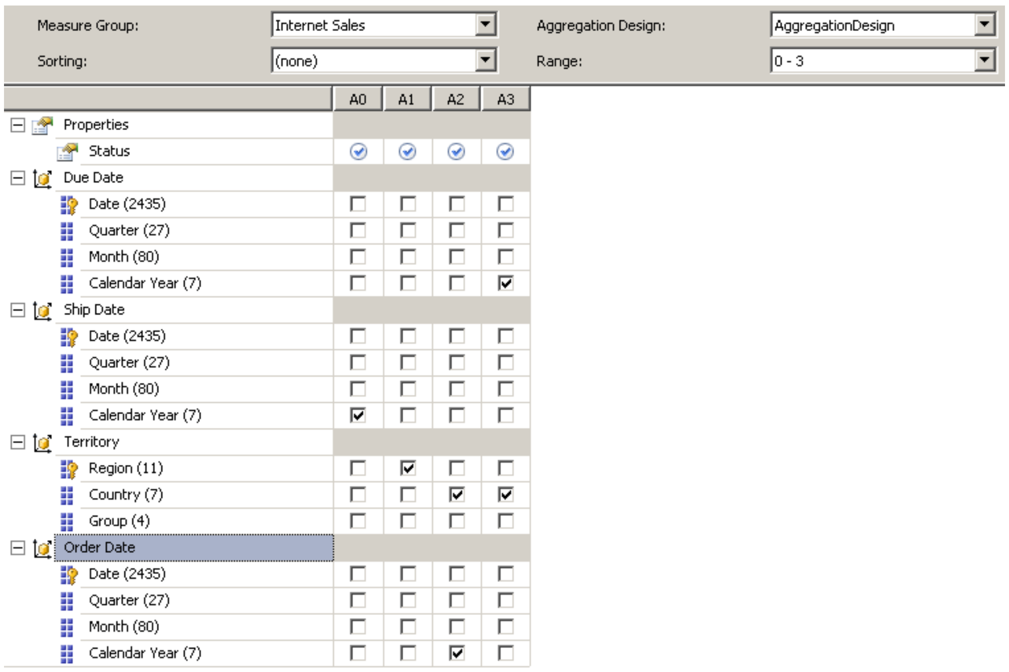
| | OrderQuantity_0 | TotalProductCost_1 | SalesAmount_2 | SalesTerritoryCountry_3 | CalendarYear_4 | COUNT_BIG_7673aff6-2445-4ef6-a4c9-7bf3d93bd42a |
|---|---|---|---|---|---|---|
| 1 | 394 | 780478.0437 | 1309047.1978 | Australia | 2005 | 394 |
| 2 | 859 | 1274424.9173 | 2154284.8835 | Australia | 2006 | 859 |
| 3 | 5335 | 1787464.4735 | 3033784.2131 | Australia | 2007 | 5335 |
| 4 | 6757 | 1532778.0736 | 2563884.29 | Australia | 2008 | 6757 |
| 5 | 47 | 88118.6533 | 146829.8074 | Canada | 2005 | 47 |
| 6 | 226 | 373871.7428 | 621602.3823 | Canada | 2006 | 226 |
| 7 | 3086 | 302837.5866 | 535784.4624 | Canada | 2007 | 3086 |
| 8 | 4261 | 383095.3782 | 673628.21 | Canada | 2008 | 4261 |
| 9 | 59 | 108108.2606 | 180571.692 | France | 2005 | 59 |
| 10 | 233 | 306757.9856 | 514942.0131 | France | 2006 | 233 |
| 11 | 2291 | 602744.6969 | 1026324.9692 | France | 2007 | 2291 |
| 12 | 2975 | 540142.0494 | 922179.04 | France | 2008 | 2975 |
| 13 | 76 | 142211.4949 | 237784.9902 | Germany | 2005 | 76 |
| 14 | 233 | 310869.6938 | 521230.8475 | Germany | 2006 | 233 |
| 15 | 2254 | 625392.7704 | 1058405.7305 | Germany | 2007 | 2254 |
| 16 | 3062 | 628467.6141 | 1076890.77 | Germany | 2008 | 3062 |
| 17 | 96 | 174339.1921 | 291590.5194 | United Kingdom | 2005 | 96 |
| 18 | 265 | 350778.4984 | 591586.854 | United Kingdom | 2006 | 265 |
| 19 | 2966 | 765690.736 | 1298248.5675 | United Kingdom | 2007 | 2966 |
| 20 | 3579 | 710413.0064 | 1210286.27 | United Kingdom | 2008 | 3579 |
| 21 | 341 | 661512.1419 | 1100549.4498 | United States | 2005 | 341 |
| 22 | 861 | 1266790.5367 | 2126696.546 | United States | 2006 | 861 |
| 23 | 8511 | 1634196.9115 | 2838512.355 | United States | 2007 | 8511 |
| 24 | 11631 | 1926309.118 | 3324031.16 | United States | 2008 | 11631 |

*Figure 57:  Indexed View for ROLAP Aggregations*

As you will learn later in Chapter 6, ROLAP storage keeps all data for a cube in a relational database, including the aggregations. However, MOLAP storage keeps all data on disk in a proprietary format. When a user executes a query against a measure group using MOLAP storage, Analysis Services executes a query that returns data much like you see in the indexed view. In essence, it queries the fact data and performs a group by operation on the designated attributes. However, when a measure group is using ROLAP storage, Analysis Services performs a relational query for ROLAP data. Each time you process the cube, each aggregation is updated to remain synchronized with the measure group data.

## Aggregation Designer

To see how the aggregation design corresponds to the aggregation itself, let's look at the Aggregation Designer. To open the Aggregation Designer, shown in Figure 58, click the **Aggregations** tab of the Cube Designer. Next, click **Advanced View** in the toolbar, and make your selections in the **Measure Group** and **Aggregation Design** drop-down lists.



| | | | |
|---|---|---|---|
| Measure Group: | Internet Sales | Aggregation Design: | AggregationDesign |
| Sorting: | (none) | Range: | 0 - 3 |

| | A0 | A1 | A2 | A3 |
|---|---|---|---|---|
| Properties | | | | |
| Status | ✓ | ✓ | ✓ | ✓ |
| Due Date | | | | |
| Date (2435) | ☐ | ☐ | ☐ | ☐ |
| Quarter (27) | ☐ | ☐ | ☐ | ☐ |
| Month (80) | ☐ | ☐ | ☐ | ☐ |
| Calendar Year (7) | ☐ | ☐ | ☐ | ☑ |
| Ship Date | | | | |
| Date (2435) | ☐ | ☐ | ☐ | ☐ |
| Quarter (27) | ☐ | ☐ | ☐ | ☐ |
| Month (80) | ☐ | ☐ | ☐ | ☐ |
| Calendar Year (7) | ☑ | ☐ | ☐ | ☐ |
| Territory | | | | |
| Region (11) | ☐ | ☑ | ☐ | ☐ |
| Country (7) | ☐ | ☐ | ☑ | ☑ |
| Group (4) | ☐ | ☐ | ☐ | ☐ |
| Order Date | | | | |
| Date (2435) | ☐ | ☐ | ☐ | ☐ |
| Quarter (27) | ☐ | ☐ | ☐ | ☐ |
| Month (80) | ☐ | ☐ | ☐ | ☐ |
| Calendar Year (7) | ☐ | ☐ | ☑ | ☐ |

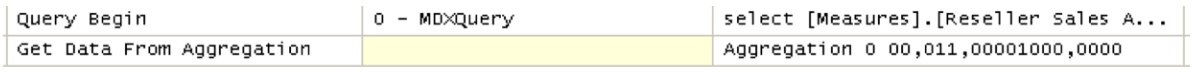
*Figure 58: Aggregation Designer*

The first aggregation is labeled A0, which is the abbreviation for Aggregation 0. The check boxes in this column identify the attributes to include in the aggregation. In this case, A0 includes only the Calendar Year aggregation. The A3 aggregation, containing Calendar Year and Country, is the aggregation corresponding to the indexed view shown in Figure 57. These four aggregations are useful when user queries ask for measures by ship date calendar year, by region, by country and order date calendar year, or by due date calendar year and country. Furthermore, a query for sales by group in the Territory dimension can use the aggregation by country to derive the group values, which Analysis Services can do much faster than by adding up fact data and grouping by country at query time.

However, if a user asks for sales by region and year, Analysis Services cannot use any of these aggregations. Instead, it will have to add up the individual transactions in the fact table and group by region and year, which takes much longer for thousands of rows than the handful of rows in these aggregations. Similarly, a query for sales by country and order date quarter cannot be answered by these aggregations, nor can a query for sales by country and order date month.

Analysis Services uses an algorithm to come up with the set of aggregations that are most useful for answering queries, trying to strike a balance between storage, processing time, and query performance, but there’s no guarantee that the aggregations it creates are the aggregations that are best for user queries. Analysis Services can use any attribute to build aggregations. Essentially, it does a cost-benefit analysis to determine where it gets the most value from the presence of an aggregation as compared to calculating an aggregated value at run time.

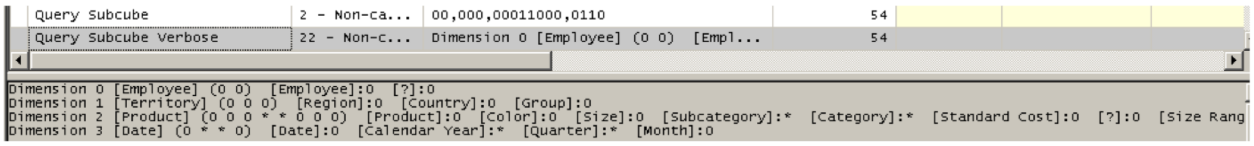
After you use the Aggregation Wizard, or even if you choose not to use the wizard, you can add new aggregations or modify existing aggregations by using the Aggregation Designer. Let's say that you've been made aware of some query performance problems. You can use SQL Server Profiler to start a trace and review the trace results to find out if aggregations are being used and to see what data is being requested. If Analysis Services can use an aggregation, you will see the event in the trace, as shown in Figure 59. You can also see which aggregation was used by matching it to the number you see in the Aggregation Designer. Furthermore, you can measure the duration of the associated Query Subcube event. If the duration is less than 500 milliseconds, the aggregation doesn't require any change.



```
Query Begin                  0 - MDXQuery      select [Measures].[Reseller Sales A...
Get Data From Aggregation                      Aggregation 0 00,011,00001000,0000
```

*Figure 59:  SQL Server Profiler Trace with Aggregation Hit*

When you find a Query Subcube event without an aggregation but with a high duration, you can use the information in the Query Subcube verbose event to determine which attributes should be in an aggregation design. In the Text field, any attribute without a 0 next to it is a candidate. For example, in Figure 60, you can see that the attributes Subcategory, Category, Calendar Year, and Quarter are each displayed with an asterisk. Therefore, you should consider creating an aggregation for Subcategory and Quarter.



```
Query Subcube             2 - Non-ca...   00,000,00011000,0110                      54
Query Subcube Verbose     22 - Non-c...   Dimension 0 [Employee] (0 0)  [Empl...   54

Dimension 0 [Employee] (0 0)  [Employee]:0  [?]:0
Dimension 1 [Territory] (0 0 0)  [Region]:0  [Country]:0  [Group]:0
Dimension 2 [Product] (0 0 0 * * 0 0 0)  [Product]:0  [Color]:0  [Size]:0  [Subcategory]:*  [Category]:*  [Standard Cost]:0  [?]:0  [Size Rang
Dimension 3 [Date] (0 * * 0)  [Date]:0  [Calendar Year]:*  [Quarter]:*  [Month]:0
```

*Figure 60:  SQL Server Profiler Trace with Aggregation Hit*

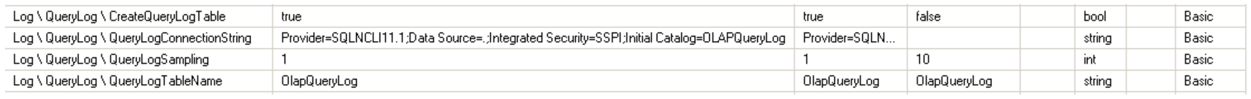
***Tip: It is not necessary to create an aggregation for attributes in higher levels of the same category. That is, you do not need to create an aggregation for Year because the lower-level Quarter aggregation suffices. Similarly, the Subcategory aggregation negates the need to create a Category aggregation.***

The toolbar in the Advanced view of the Aggregation Designer allows you to create a new aggregation design, either by starting with an empty aggregation design or by creating a copy of an existing design. You can do this if you don't have an existing aggregation design on the partition accessed by the query, or you want to replace the existing aggregation design. Simply click **New Aggregation Design** on the toolbar to start a new design. Then you can click **New Aggregation** on the toolbar to add a new aggregation and select the check boxes for the lowest level attribute of a hierarchy necessary to support user queries. In the previous example, the selection includes only the Subcategory and Quarter attributes.

## Usage-Based Optimization

The problem with the Aggregation Wizard is that the aggregation design process is based on the counts of dimension members relative to fact records. In practice, there's no point in creating aggregations if they don't help queries that are actually executing. Fortunately, you can use Usage-Based Optimization (UBO) to find out which queries are running and design aggregations to better support those queries.

To work with UBO, you first need to configure the query log on the Analysis Server. Open **SQL Server Management Studio**, connect to Analysis Services, right-click the server node, and click **Properties**. There are four properties to configure, as shown in Figure 61.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Log \ QueryLog \ CreateQueryLogTable | true | true | false | | bool | | Basic |
| Log \ QueryLog \ QueryLogConnectionString | Provider=SQLNCLI11.1;Data Source=.;Integrated Security=SSPI;Initial Catalog=OLAPQueryLog | Provider=SQLN... | | | string | | Basic |
| Log \ QueryLog \ QueryLogSampling | 1 | 1 | 10 | | int | | Basic |
| Log \ QueryLog \ QueryLogTableName | OlapQueryLog | OlapQueryLog | OlapQueryLog | | string | | Basic |

*Figure 61:  SQL Server Profiler Trace with Aggregation Hit*

By default, the query logging mechanism is not enabled. To enable it, set the **CreateQueryLogTable** property to *true*. For the **QueryLogConnectionString** property, you must provide a connection string for a SQL Server database in which you want to store the log. In addition, you need to specify a sampling rate. For initial analysis of queries, you should change the **QueryLogSampling** value to 1 to capture every query. Lastly, you can customize the name of the table in which to store the query log data in the **QueryLogTableName** property. The default table name is **OlapQueryLog**.

***Tip: When you have finished capturing log data, you can clear the connection string to discontinue logging. If you do keep the query logging active, Analysis Services deletes records from the table when you make structural changes that impact aggregation design, such as changing the number of measures in a measure group, changing the number of dimensions associated with a measure group, or changing the number of attributes defined for a dimension.***

After enabling the query log, you can run standard reports or just let users browse the cube using their favorite tools for a period of time, such as a week or a month. During that time, usage information about queries is added to the query log table. This table tracks queries by database, the partition queried, and user. It also records the dataset used, the start time, and the query duration. The dataset is in a format that matches the Query Subcube Event that you see in Profiler.

When you're ready to design aggregations based on usage, launch the Usage Based Optimization wizard. In SSDT, click the second button on the toolbar of the **Aggregations** tab in the Cube Designer. After you launch the wizard, you can filter the contents of the query log by date, user, or most frequent queries. If you need to filter by other values, you can execute a T-SQL script to delete rows from the query log manually.

As you step through the wizard, you can see information about the queries, such as the attributes used, the frequency of that combination of attributes in queries, and the average duration of queries. You can't change any information here, but you can sort it and see what kind of activity has been captured. The remaining steps in the Usage Based Optimization Wizard are very similar to the steps in the Aggregation Wizard.

## Perspectives

By default, all objects that you create in a cube are visible to each user, unless you apply security to dimensions, dimensions members, or cells as described in Chapter 6. When a cube contains many objects, you can simplify the cube navigation experience for the user by adding a perspective to the cube. That way, the user sees only a subset of the cube when browsing the metadata to select objects for a query, as shown in Figure 62.

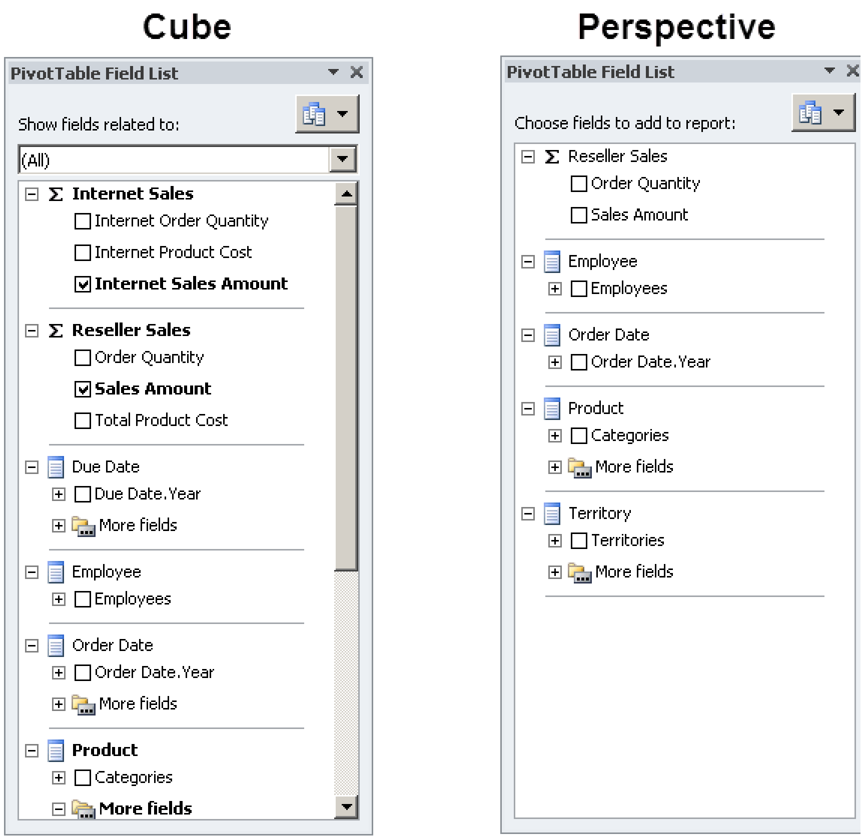
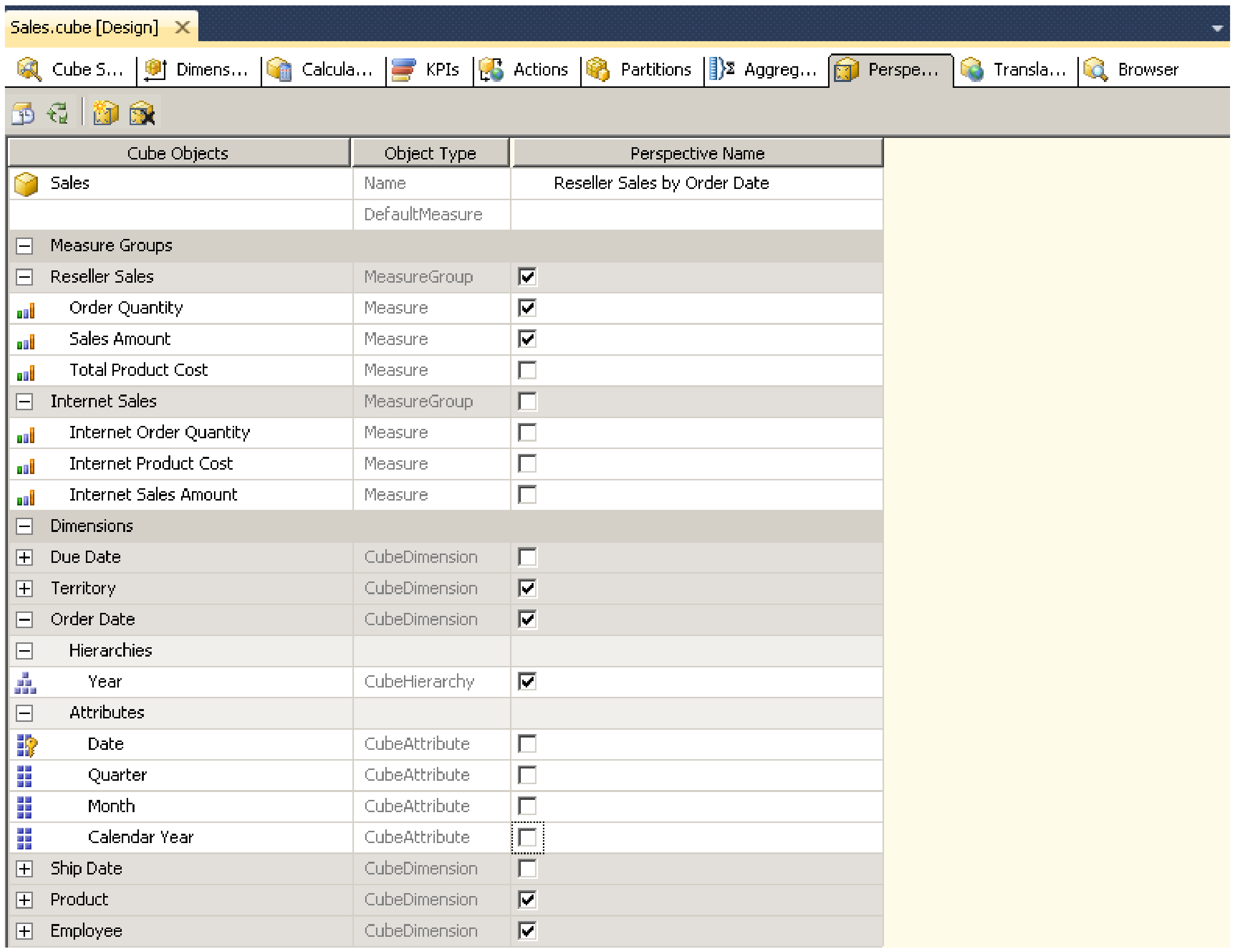


*Figure 62: Browsing a Cube versus a Perspective*

To create a perspective, open the **Perspectives** tab in the Cube Designer, and then click **New Perspective** on the toolbar. You can assign a name to the new perspective, and then clear or select check boxes to identify the objects to exclude or include respectively, as shown in Figure 63.



| Cube Objects | Object Type | Perspective Name |
|---|---|---|
| Sales | Name | Reseller Sales by Order Date |
| | DefaultMeasure | |
| Measure Groups | | |
| Reseller Sales | MeasureGroup | ✓ |
| Order Quantity | Measure | ✓ |
| Sales Amount | Measure | ✓ |
| Total Product Cost | Measure | ☐ |
| Internet Sales | MeasureGroup | ☐ |
| Internet Order Quantity | Measure | ☐ |
| Internet Product Cost | Measure | ☐ |
| Internet Sales Amount | Measure | ☐ |
| Dimensions | | |
| Due Date | CubeDimension | ☐ |
| Territory | CubeDimension | ✓ |
| Order Date | CubeDimension | ✓ |
| Hierarchies | | |
| Year | CubeHierarchy | ✓ |
| Attributes | | |
| Date | CubeAttribute | ☐ |
| Quarter | CubeAttribute | ☐ |
| Month | CubeAttribute | ☐ |
| Calendar Year | CubeAttribute | ☐ |
| Ship Date | CubeDimension | ☐ |
| Product | CubeDimension | ✓ |
| Employee | CubeDimension | ✓ |

*Figure 63: New Perspective*

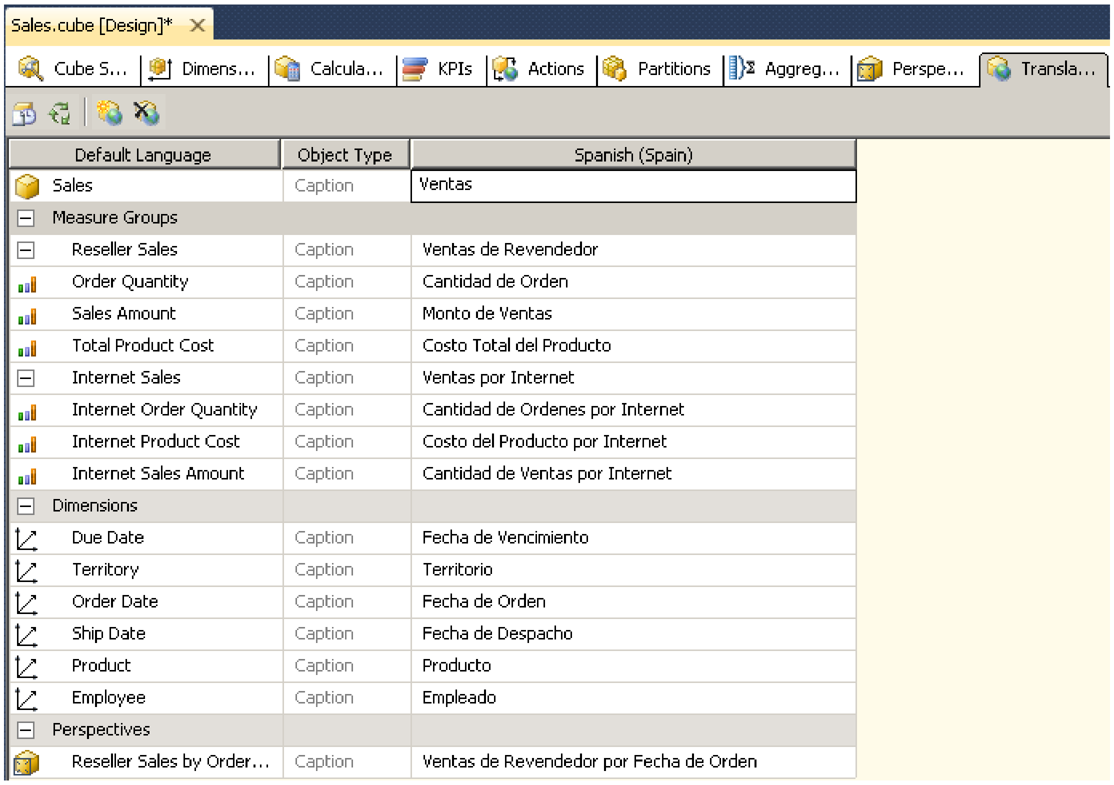
You can include any of the following object types in a perspective:

- Measure group
- Measure
- Calculation
- Key performance indicator
- Dimension
- Hierarchy
- Attribute

***Note: A perspective cannot be used as a security measure. Its sole purpose is to simplify the user interface by displaying a subset of objects in a cube. To prevent users from viewing sensitive data, you must configure security as described in Chapter 6.***

## Translations

Just as you can display translations for dimensions as described in Chapter 3, "Developing dimensions," you can also set up translations for a cube. However, there is no option to retrieve the translation from a relational source. Instead, you must manually type the translated caption for the cube, measure groups, measures, dimensions, and perspectives. To add the translations, open the **Translations** tab of the Cube Designer, click **New Translation** on the toolbar, and type the applicable caption for each object, as shown in Figure 64.



| Default Language | Object Type | Spanish (Spain) |
|---|---|---|
| Sales | Caption | Ventas |
| Measure Groups | | |
| Reseller Sales | Caption | Ventas de Revendedor |
| Order Quantity | Caption | Cantidad de Orden |
| Sales Amount | Caption | Monto de Ventas |
| Total Product Cost | Caption | Costo Total del Producto |
| Internet Sales | Caption | Ventas por Internet |
| Internet Order Quantity | Caption | Cantidad de Ordenes por Internet |
| Internet Product Cost | Caption | Costo del Producto por Internet |
| Internet Sales Amount | Caption | Cantidad de Ventas por Internet |
| Dimensions | | |
| Due Date | Caption | Fecha de Vencimiento |
| Territory | Caption | Territorio |
| Order Date | Caption | Fecha de Orden |
| Ship Date | Caption | Fecha de Despacho |
| Product | Caption | Producto |
| Employee | Caption | Empleado |
| Perspectives | | |
| Reseller Sales by Order... | Caption | Ventas de Revendedor por Fecha de Orden |

*Figure 64: Translation Captions in a Cube*

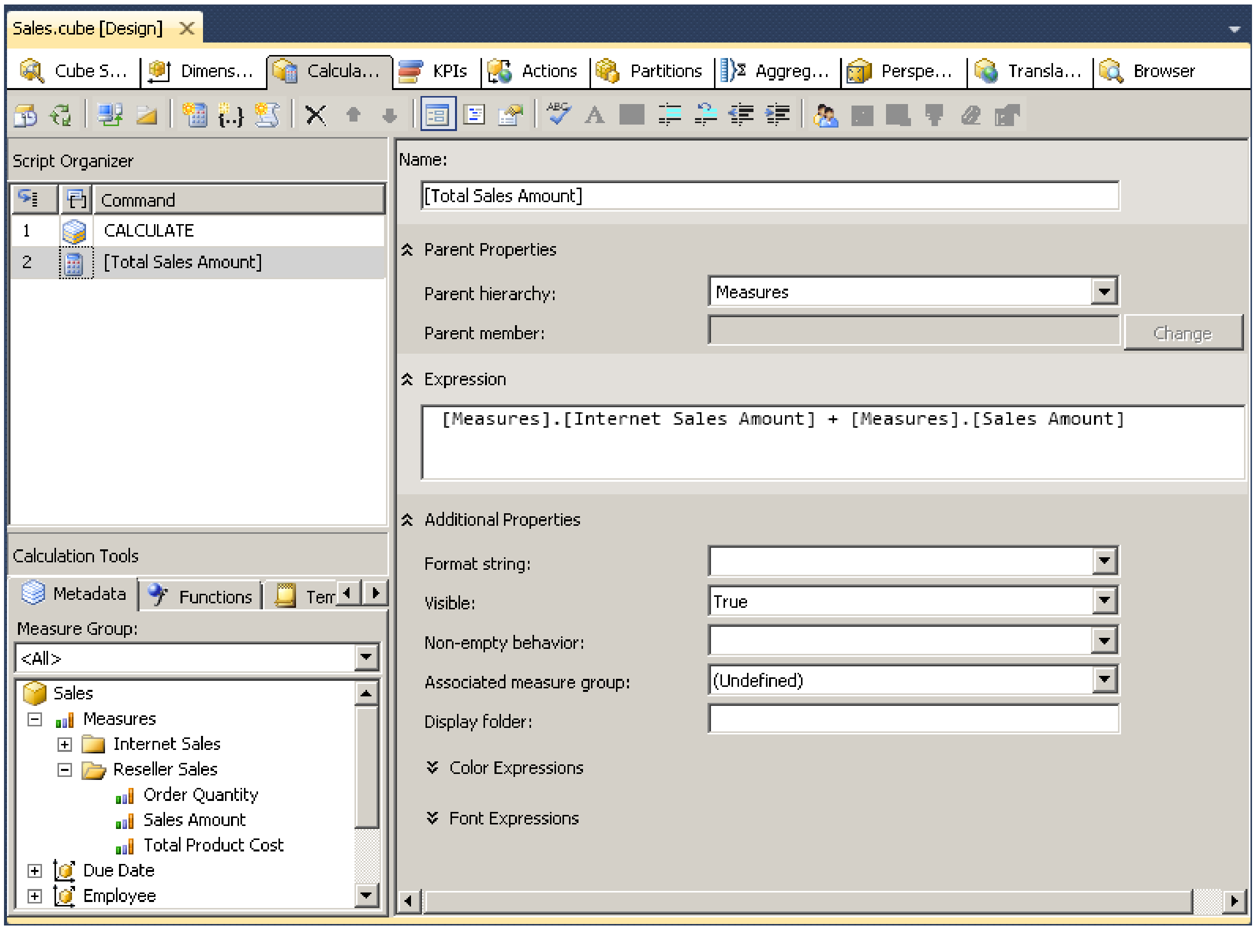
# Chapter 5  Enhancing Cubes with MDX

One of the benefits of using an Analysis Services database for reporting and analysis (instead of using a relational database) is the ability to store business logic as calculations directly in the cube. That way, you can ensure that a particular business metric is calculated the same way every time, no matter who is asking to view the metric and no matter what skill level they have. To store business logic in the cube, you use MDX, which is the abbreviation for multidimensional expression language. MDX is also a query language that presentation layer tools use to retrieve data from a cube.

I'll begin this chapter by explaining how to add basic calculations to the cube. Users can access these calculations just like any other measure.  The calculation designer includes several tools to help you create calculations, ranging from finding the right names of objects to selecting functions or templates for building calculations. In this chapter, you learn how to create simple calculations and how to use expressions to change the appearance of calculation results, so that users can tell at a glance whether a number is above or below a specified threshold. In addition, this chapter shows you how to use expressions to add custom members and named sets to a dimension. You will also learn about the more specialized types of expressions for key performance indicators to track progress towards goals and for actions to add interactivity to cube browsing.

## Calculated Members

To create a calculated member, open the **Calculation** tab of the Cube Designer in SSDT and click **New Calculated Member** in the toolbar, the fifth button from the left. The Form View asks for the information that we need based on the type of calculation we're creating. For a calculated member, you need to supply a name and an expression, and set the **Visible** property at minimum, as shown in Figure 65. The name needs to be enclosed in braces if it has spaces in it or starts with a numeric value, and the expression must be a valid MDX expression.



| | Command |
|---|---|
| 1 | CALCULATE |
| 2 | [Total Sales Amount] |

Measure Group: <All>

Name: [Total Sales Amount]

Parent Properties

Parent hierarchy: Measures

Parent member:

Expression

```
[Measures].[Internet Sales Amount] + [Measures].[Sales Amount]
```

Additional Properties

Format string:

Visible: True

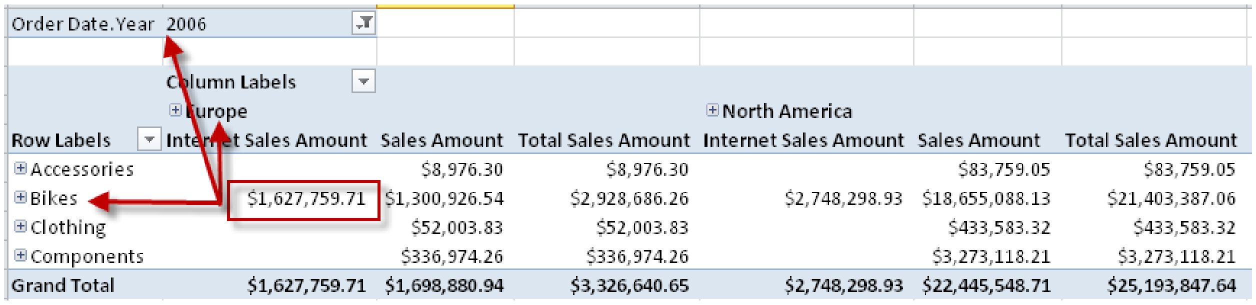
Non-empty behavior:

Associated measure group: (Undefined)

Display folder:

*Figure 65: New Calculated Member*

In this example, a simple addition operation combines the results of evaluating two measures, Internet Sales Amount and Sales Amount. For each cell intersection of rows and columns in a query, Analysis Services evaluates each portion of the expression individually and adds the results. That is, Internet Sales Amount is evaluated in the context of the dimensions on the current row, the current column, and the current filter, as shown in Figure 66, and the same is true for Sales Amount. Then, Analysis Services adds the results to produce the value that is displayed for Total Sales Amount having the same combination of row, column, and filter members.



Order Date.Year 2006

| | Column Labels | | | | | |
|---|---|---|---|---|---|---|
| | Europe | | | North America | | |
| Row Labels | Internet Sales Amount | Sales Amount | Total Sales Amount | Internet Sales Amount | Sales Amount | Total Sales Amount |
| Accessories | | $8,976.30 | $8,976.30 | | $83,759.05 | $83,759.05 |
| Bikes | $1,627,759.71 | $1,300,926.54 | $2,928,686.26 | $2,748,298.93 | $18,655,088.13 | $21,403,387.06 |
| Clothing | | $52,003.83 | $52,003.83 | | $433,583.32 | $433,583.32 |
| Components | | $336,974.26 | $336,974.26 | | $3,273,118.21 | $3,273,118.21 |
| Grand Total | $1,627,759.71 | $1,698,880.94 | $3,326,640.65 | $2,748,298.93 | $22,445,548.71 | $25,193,847.64 |

*Figure 66: New Calculated Member*

## Calculated Member Properties

A calculation also has several properties, many of which are similar to measures:

- **Format String**. Just as you can set a format string for a measure, you can select a built-in string or type in a custom string for a calculated member.
- **Visible**. Rather than try to build one complex calculation, it's usually better for troubleshooting purposes to break it down into separate components that you hide by setting this property to False. You can then use the individual calculated members in the calculated member that is visible.
- **Non-Empty Behavior**. For this property, you specify one or more measures that Analysis Services should evaluate to determine whether the calculated member is empty. If you select multiple measures, all measures must be empty for Analysis Services to consider the calculated member to be empty also. On the other hand, if you do not configure this property, Analysis Services evaluates the calculated member to determine if it's empty, which may increase query response time.
- **Associated Measure Group**. If you select a measure group, the calculated member will appear in the measure group's folder in the metadata tree. Otherwise, it is displayed in a separate section of the metadata tree along with other unassociated calculated members.
- **Display Folder**. Another option for organizing calculated members is to use display folders. You can select an existing folder in the drop-down list or type the name of a new folder to create it.
- **Color Expressions**. You can define expressions to determine what font or cell color to display in the presentation tool when Analysis Services evaluates the calculated member.
- **Font Expressions**. You can also define expressions to determine the font name, size, or properties when the calculated member is displayed in the presentation tool.

***Note: Presentation tools vary in their support of color and font expressions. Be sure to test a calculated member in each presentation tool that you intend to use and set user expectations accordingly.***

Each calculated member becomes part of the cube's MDX script. The MDX script is a set of commands and expressions that represent all of the calculations for the entire cube, including the named sets and key performance indicators that I explain later in this chapter. Using the MDX script, you can not only create calculations to coexist with the measures that represent aggregated values from the fact table, but you can also design some very complex behavior into the cube, such as changing values at particular cell intersections. However, working with the MDX script extensively requires a deep understanding of MDX and is beyond the scope of this book.

***Tip: To learn more about MDX, see [Stairway to MDX](), an online tutorial. You can learn about working with MDX script at [MDX Scripting Fundamentals]() at MSDN.***

## Calculation Tools

To help you build an MDX expression, the Calculation Designer includes several tools:

- Metadata
- Functions
- Templates

In the bottom left corner of the Calculation Designer is the Calculation Tools pane, as shown in Figure 67. The first tab of this pane displays **Metadata**, which contains the tree structure of the cube and its dimensions. At the top of the tree is the list of measures organized by measure group in addition to calculated members. Dimensions appear below the measures. In the metadata tree, you can access any type of object—the dimension itself, a hierarchy, a level of a hierarchy, or an individual dimension member.

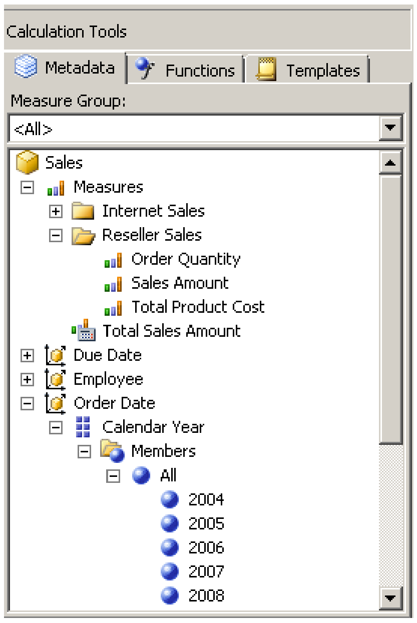


*Figure 67: Metadata in Calculation Tools Pane*

To use the metadata tree, right-click the object to add to your expression, and click **Copy**. Next, right-click in the **Expression** box, and click **Paste**. You can type any mathematical operators that you need in the expression, and then continue selecting items from the metadata tree.

The **Functions** tab in the Calculation Tools pane contains several folders, which organize MDX functions by category. Figure 68 shows some of the navigation functions, which are very commonly used in MDX when you have hierarchies and want to do comparisons over time or percentage of total calculations.

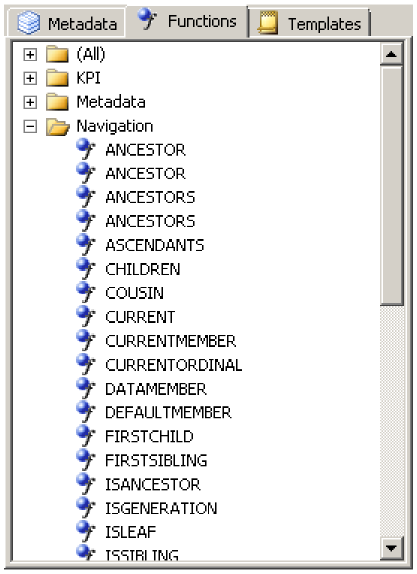


*Figure 68: Functions in Calculation Tools Pane*

It's important to understand what object type a function returns so that you can use it properly in an expression. For example, the [Children](#) function returns a set of members. Let's put this in context by looking at the Categories hierarchy, shown in Figure 69.

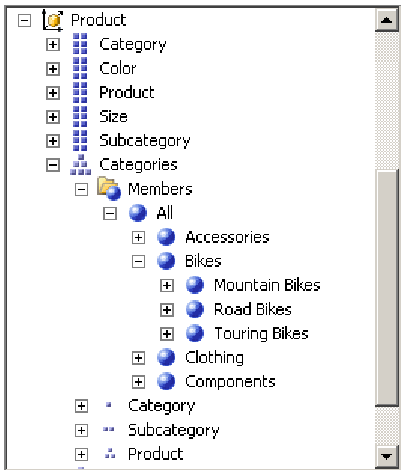


*Figure 69: Dimension Members in a User-Defined Hierarchy*

At the top of the hierarchy is the All member, which is the parent of the members that exist on the level below: Accessories, Bikes, Clothing, and Components. You can use the [Parent](#) function with any of these members to return that All member, like this:

```
[Product].[Categories].[Accessories].Parent
```

Likewise, you can use the **Children** function to get the set of members below a specified member. When you use this function with the Bikes member, the result is a set containing the Mountain Bikes, Road Bikes, and Touring Bikes members:

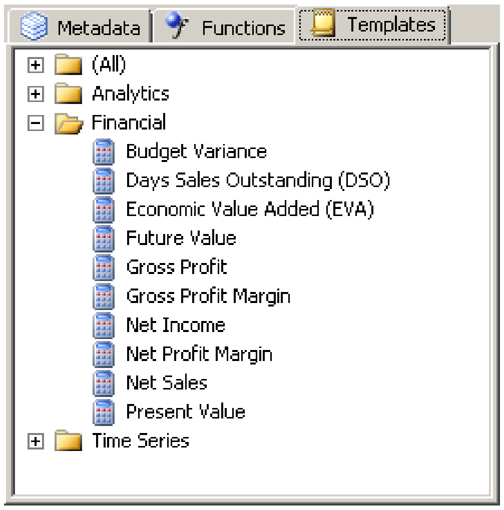
```
[Product].[Categories].[Bikes].Children
```

You can also use navigation functions to move forward and backward along the same level. For example, the NextMember function returns Road Bikes:

```
[Product].[Categories].[Mountain Bikes].NextMember
```

Likewise, using the PrevMember function with Touring Bikes also returns Road Bikes. It's more common to use **NextMember** and **PrevMember** functions with a date dimension, but the functions work with any type of dimension. To summarize, an expression for calculated members consists of references to measures or members by using a direct reference (such as `[Product].[Categories].[Mountain Bikes]`) or a function to get a relative reference (such as `[Product].[Categories].[Accessories].Parent`). You can also perform mathematical operations on these references.

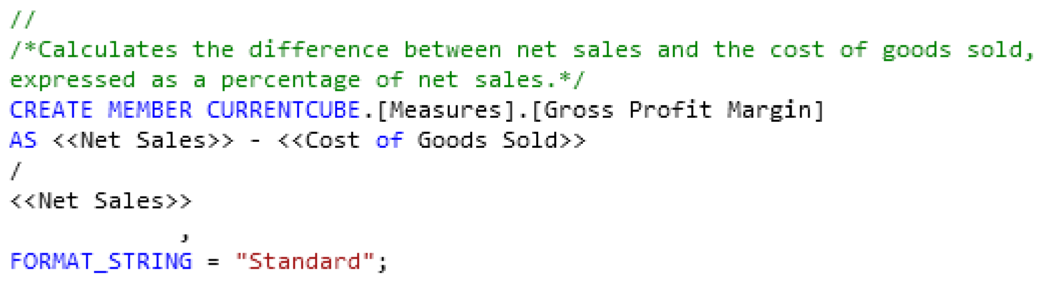
The third tab in the Calculation Tools pane is **Templates**. This tab provides sample expressions that you can use for many common types of calculations, as shown in Figure 70. For example, in a sales cube, you might have a calculated measure for gross profit margin. To use the template, you must first switch to Script View by using the twelfth button on the toolbar. Position the pointer below existing expressions, right-click the template, and then click **Add To Template**.



- (All)
- Analytics
- Financial
  - Budget Variance
  - Days Sales Outstanding (DSO)
  - Economic Value Added (EVA)
  - Future Value
  - Gross Profit
  - Gross Profit Margin
  - Net Income
  - Net Profit Margin
  - Net Sales
  - Present Value
- Time Series

*Figure 70: Templates for Expressions*

Unlike selecting an object from the metadata tree, the template adds the complete script command to create a calculation as shown in Figure 71. The expression itself contains tokens that we need to replace with actual references from our cube. You can do that in Script View or you can toggle back to Form View (the eleventh button in the toolbar).



```
//
/*Calculates the difference between net sales and the cost of goods sold,
expressed as a percentage of net sales.*/
CREATE MEMBER CURRENTCUBE.[Measures].[Gross Profit Margin]
AS <<Net Sales>> - <<Cost of Goods Sold>>
/
<<Net Sales>>
            ,
FORMAT_STRING = "Standard";
```

*Figure 71: Templates for Expressions*

Either way, the completed expression requires the sales amount and cost measures. In addition, you must add parentheses to the numerator expression to get the right order of operations and change the format string. That is, you must subtract cost from sales to get the margin before performing the division operation, like this:

```
CREATE MEMBER CURRENTCUBE.[Measures].[Gross Profit Margin]
AS ([Measures].[Sales Amount] - [Measures].[Total Product Cost])
/
[Measures].[Sales Amount]
,
FORMAT_STRING = "Percent";
```

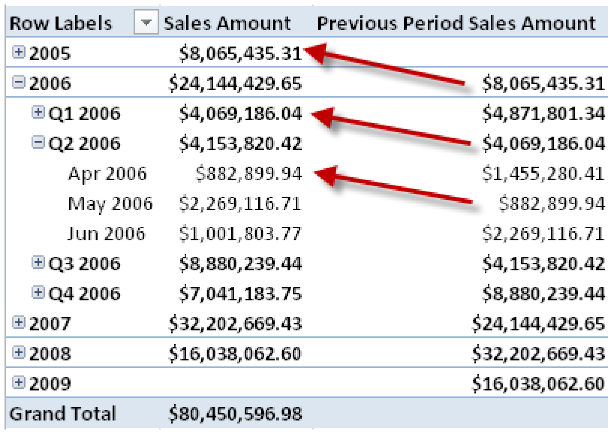
## Tuple Expressions

A common way to use navigation functions is to retrieve values for a different time period. For example, to retrieve sales for a previous period, you can combine the sales amount measure with the **PrevMember** function in conjunction with a date hierarchy like this:

```
([Measures].[Sales Amount], [Order Date].[Year].PrevMember)
```

This type of expression is known as a **tuple expression**. It is a coordinate that tells Analysis Services where to find information in the cube. A tuple can contain members only, which is why it's important to know what a function returns. Because **PrevMember** returns a member, it's valid for the tuple expression in the previous example.

By using a navigation function in a tuple expression, you can reposition Analysis Services away from a current member to a different member by moving forward or backward in the same level or up or down within a hierarchy. For example, in Figure 72, the Previous Period Sales Amount (based on the tuple expression defined previously), you can see the value for 2006 matches the Sales Amount value for 2005, which is the previous member on the Year level of the Year user-defined hierarchy of the Order Date dimension. Similarly, the Previous Period Sales Amount value for Q2 2006 is the Sales Amount value for Q1 2006, the previous member on the Quarter level of the hierarchy. The same is true at the month level, where the May 2006 value for Previous Period Sales Amount matches the April 2006 value for Sales Amount.



| Row Labels | Sales Amount | Previous Period Sales Amount |
|---|---|---|
| ⊞2005 | $8,065,435.31 | |
| ⊟2006 | $24,144,429.65 | $8,065,435.31 |
| ⊞Q1 2006 | $4,069,186.04 | $4,871,801.34 |
| ⊟Q2 2006 | $4,153,820.42 | $4,069,186.04 |
| Apr 2006 | $882,899.94 | $1,455,280.41 |
| May 2006 | $2,269,116.71 | $882,899.94 |
| Jun 2006 | $1,001,803.77 | $2,269,116.71 |
| ⊞Q3 2006 | $8,880,239.44 | $4,153,820.42 |
| ⊞Q4 2006 | $7,041,183.75 | $8,880,239.44 |
| ⊞2007 | $32,202,669.43 | $24,144,429.65 |
| ⊞2008 | $16,038,062.60 | $32,202,669.43 |
| ⊞2009 | | $16,038,062.60 |
| Grand Total | $80,450,596.98 | |

*Figure 72: Effect of PrevMember function in Tuple Expression*

## Color and Font Expressions

Calculated members have color and font properties that are not available for measures. These color and font properties can be based on expressions, and are configured at the bottom of the Form View. However, if you know the correct syntax, you can add these expressions directly to the MDX script.

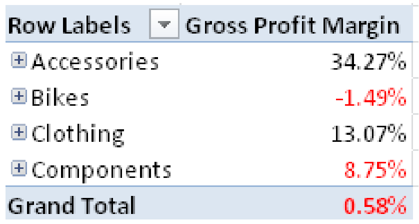
In Form View, expand the applicable section. For example, in the Color Expressions, you have the option to configure Fore Color to set the font color or Back Color to set the fill color of the cell containing a value. In the Font Expressions, you can set Font Name or Font Size. You can also set Font Flags when you want to control whether to display a value using bold or italic font styling.

Typically, you use conditional expressions for colors and fonts. A conditional expression uses the IIF function, which has three arguments. The first argument is a Boolean expression that must resolve as true or false. Usually you compare a value to a threshold here. The second argument is the value returned by the function when the Boolean expression is true, while the third argument is the value returned when the Boolean expression is false, like this:

```
iif([Measures].[Gross Profit Margin]<.1, 255 /*Red*/, 0 /*Black*/)
```

***Tip: You can use the color or font picker that is displayed to the right of the expression box to set the appropriate code in your expression. For example, you can select the color red in the picker and the value 255 appears in your expression along with a commented string, /*Red*/, as a reminder of the value's meaning.***

You should then test the result in the presentation tools that users will have, such as Microsoft Excel, as shown in Figure 73. Here, you can see the red font is displayed only wherever the gross profit margin is less than 10%.

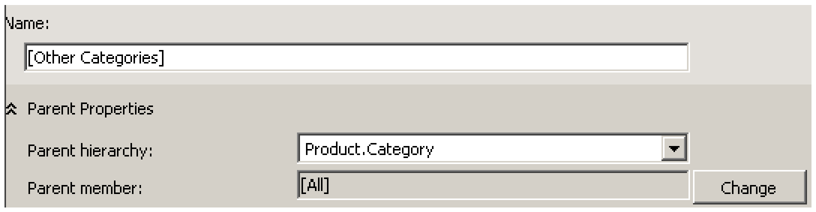


| Row Labels | Gross Profit Margin |
|---|---|
| ⊞Accessories | 34.27% |
| ⊞Bikes | -1.49% |
| ⊞Clothing | 13.07% |
| ⊞Components | 8.75% |
| Grand Total | 0.58% |

*Figure 73: Conditional Color Expression*

## Custom Members

Sometimes you need to create calculations that affect how measures get computed, but are not themselves measures. These are calculated members that are part of a dimension, and are also known as custom members. Let's say that you want to support analysis of one product category relative to all other product categories. You can create a calculated member to include a total of all product categories except for Bikes. When you create a calculated member, as shown in Figure 74, you assign a name for the member that is unique within the parent hierarchy and define its parent member.



Name:
[Other Categories]

Parent Properties

Parent hierarchy: Product.Category

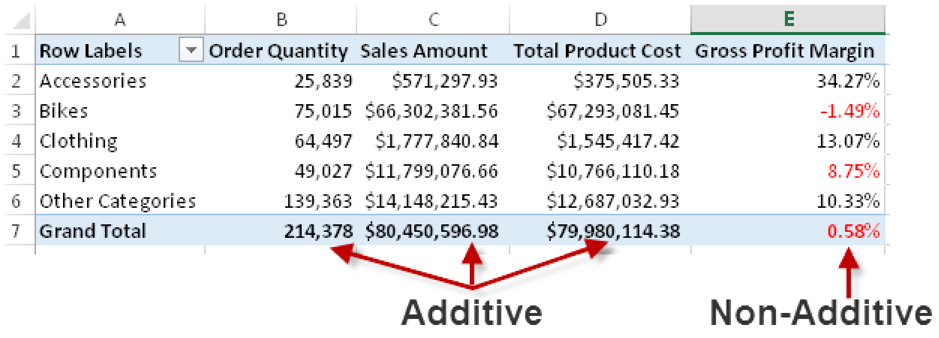
Parent member: [All]

*Figure 74: Name and Parent Properties for Calculated Member*

After that, you define the expression for the calculated member. In this example, the calculated member's expression uses the Aggregate function, but any function for aggregation is permissible here. That is, you can use Sum, Avg, Min, Max, DistinctCount, or Count instead.

```
Aggregate(
   {[Product].[Category].[Accessories], [Product].[Category].[Clothing],
    [Product].[Category].[Components]}
)
```

By using the **Aggregate** function in the expression, you can ensure the correct values are computed for any measure that you use in a tuple expression with the calculated member. Some measures are additive, like sales and costs, while other measures are non-additive, like Gross Profit Margin. With non-additive measures, Analysis Services must sum the component parts of the expression first. With Gross Profit Margin, it needs to sum the numerator separately from the denominator, and then perform the division operation last. The **Aggregate** function ensures the correct order of operations, as shown in Figure 75.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Row Labels | Order Quantity | Sales Amount | Total Product Cost | Gross Profit Margin |
| 2 | Accessories | 25,839 | $571,297.93 | $375,505.33 | 34.27% |
| 3 | Bikes | 75,015 | $66,302,381.56 | $67,293,081.45 | -1.49% |
| 4 | Clothing | 64,497 | $1,777,840.84 | $1,545,417.42 | 13.07% |
| 5 | Components | 49,027 | $11,799,076.66 | $10,766,110.18 | 8.75% |
| 6 | Other Categories | 139,363 | $14,148,215.43 | $12,687,032.93 | 10.33% |
| 7 | Grand Total | 214,378 | $80,450,596.98 | $79,980,114.38 | 0.58% |

*Figure 75: Aggregate Function Results*

The **Aggregation** function, like the other aggregation functions, requires a set as its first argument. A **set** is a collection of members from the same dimension and hierarchy. Notice that in the previous expression, the set is enclosed in braces and consists of specific members—Accessories, Clothing, and Components. In other words, the set definition explicitly lists every member of the Category hierarchy except Bikes and the All member.

Because the cube often has different types of measures, it is likely that you will need to use a conditional expression to define the format string. A CASE statement might be easier to read than a nested IIF function when there are multiple possible conditions, like this:

```
case
when [Measures].CurrentMember is [Measures].[Gross Profit Margin]
then "Percent"
when [Measures].CurrentMember is [Measures].[Order Quantity] or
     [Measures].CurrentMember is [Measures].[Internet Order Quantity]
then "#,#"
else "Currency"
end
```

You can switch to Script View to more easily work with a complex conditional expression. To do this, click the twelfth button on the Calculation Designer toolbar. Another property that you might need to set in Script View is **Language**. Sometimes Excel or other presentation layer tools are unable to properly interpret a format string, so the language property is used to give it a hint. The complete calculated member definition in Script View looks like this:

```
CREATE MEMBER CURRENTCUBE.[Product].[Category].[All].[Other Categories]
 AS Aggregate(
   {[Product].[Category].[Accessories], [Product].[Category].[Clothing],
[Product].[Category].[Components]}
),
FORMAT_STRING = case
when [Measures].CurrentMember is [Measures].[Gross Profit Margin]
then "Percent"
when [Measures].CurrentMember is [Measures].[Order Quantity] or
     [Measures].CurrentMember is [Measures].[Internet Order Quantity]
then "#,#"
else "Currency"
end
,
VISIBLE = 1,
Language = 1033;
```

***Tip: The language identifier for United States English is 1033. If you need to use a different language identifier, refer to http://msdn.microsoft.com/en-us/goglobal/bb964664.aspx to locate the correct value.***

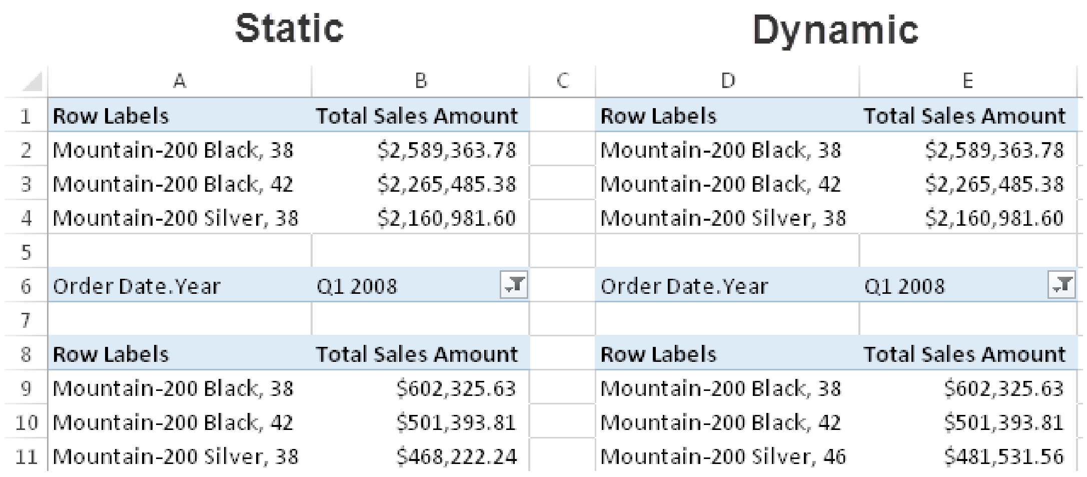
## Named Sets

You can also use the Calculation Designer to develop named sets. A **named set** is a special type of calculation that returns a collection of dimension members, rather than a specific value like a calculated measure. You can create a named set by defining a list of members as described in the previous example, or you can use a set function to generate a set at query time. For example, you can use the TopCount function to produce a set of the three products that have the highest sales, like this:

```
TopCount([Product].[Product].[Product].Members, 3,[Measures].[Total Sales
Amount])
```

The first argument of the **TopCount** function requires a set. In this case, the expression for the first argument uses the Members function to get all the members that exist on the Product level of the Product hierarchy of the Product dimension. The second argument specifies how many members to include in the final set. The third argument specifies the measure for which Analysis Services needs to perform a descending sort of the members. In other words, this function takes the set of products, finds the total sales amount for each of them, sorts the products in descending order, and then returns the first three members of the resulting set.

When you create a named set, you specify whether you want the contents of the named set to remain constant, or the contents to change based on the context of the current query. In the **Additional Properties** section of the form view for a named set, select **Static** for a constant set or **Dynamic** for a set that adjusts to the query context in the **Type** drop-down. Figure 76 shows the effect of changing the year filter on each type of named set.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| | Static | | | Dynamic | |
| 1 | Row Labels | Total Sales Amount | | Row Labels | Total Sales Amount |
| 2 | Mountain-200 Black, 38 | $2,589,363.78 | | Mountain-200 Black, 38 | $2,589,363.78 |
| 3 | Mountain-200 Black, 42 | $2,265,485.38 | | Mountain-200 Black, 42 | $2,265,485.38 |
| 4 | Mountain-200 Silver, 38 | $2,160,981.60 | | Mountain-200 Silver, 38 | $2,160,981.60 |
| 5 | | | | | |
| 6 | Order Date.Year | Q1 2008 | | Order Date.Year | Q1 2008 |
| 7 | | | | | |
| 8 | Row Labels | Total Sales Amount | | Row Labels | Total Sales Amount |
| 9 | Mountain-200 Black, 38 | $602,325.63 | | Mountain-200 Black, 38 | $602,325.63 |
| 10 | Mountain-200 Black, 42 | $501,393.81 | | Mountain-200 Black, 42 | $501,393.81 |
| 11 | Mountain-200 Silver, 38 | $468,222.24 | | Mountain-200 Silver, 46 | $481,531.56 |

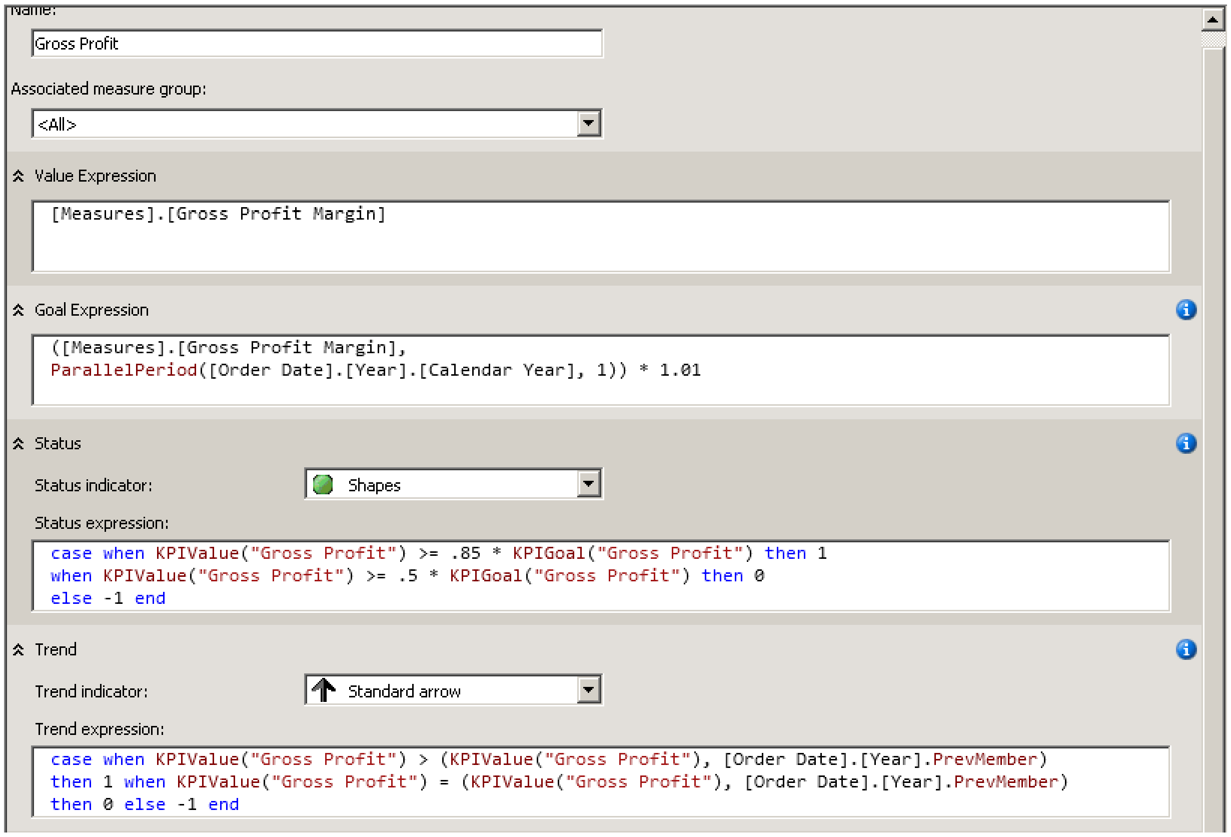
*Figure 76: Static versus Dynamic Named Sets*

On the left side of Figure 76, the pivot tables display a static named set while the pivot tables on the right side of the figure display a dynamic named set. The pivot tables on the top row are identical because no filters are applied to the static and dynamic pivot tables. They include the same top three products and the same values. However, the bottom row illustrates the difference between static and dynamic named sets when a filter, such as Q1 2008, is applied. On the left, the static named set includes the same three products that are displayed in the unfiltered pivot table above it, but the values in the Total Sales Amount column are applicable to Q1 2008 for those products. On the right, the dynamic named set displays a different set of three products. These three products are the top three selling products within the filtered time frame, Q1 2008, and the Total Sales Amount column displays the sales for those three products in that quarter. Thus, a static named set retains the same members regardless of filters in the query, whereas a dynamic named set includes different members depending on the query filters.

## Key Performance Indicators

Another type of calculation that you can add to the cube is a **key performance indicator** (KPI). Technically, a KPI is not a single calculation, but a set of calculations that quantitatively describe how a value compares to a goal. In addition, a KPI includes one symbol to represent progress toward a goal and another symbol to represent the direction the KPI value is trending at a selected point in time.

Although KPIs are calculations, you use the KPI Designer to define each KPI rather than the Calculation Designer. Figure 77 shows the set of calculations for a Gross Profit Margin KPI:



Gross Profit

Associated measure group:

<All>

Value Expression

```
[Measures].[Gross Profit Margin]
```

Goal Expression

```
([Measures].[Gross Profit Margin],
ParallelPeriod([Order Date].[Year].[Calendar Year], 1)) * 1.01
```

Status

Status indicator: Shapes

Status expression:

```
case when KPIValue("Gross Profit") >= .85 * KPIGoal("Gross Profit") then 1
when KPIValue("Gross Profit") >= .5 * KPIGoal("Gross Profit") then 0
else -1 end
```

Trend

Trend indicator: Standard arrow

Trend expression:

```
case when KPIValue("Gross Profit") > (KPIValue("Gross Profit"), [Order Date].[Year].PrevMember)
then 1 when KPIValue("Gross Profit") = (KPIValue("Gross Profit"), [Order Date].[Year].PrevMember)
then 0 else -1 end
```

*Figure 77: Key Performance Indicator Form View*

After you click the **New KPI** button in the toolbar, you define a KPI with four separate expressions: value, goal, status, and trend. Let's look at each of them separately.

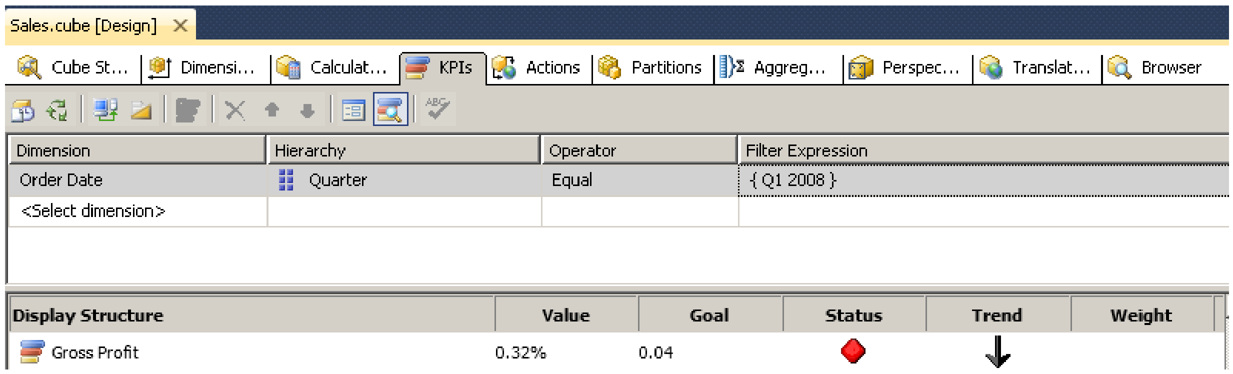
- **Value**. This expression represents the value to evaluate against a goal. It represents the current state of some metric. For example, to monitor gross profit margin, the value expression can simply be the measure already created for Gross Profit Margin. If the measure did not exist, you could use the expression to calculate gross profit margin here, although for troubleshooting purposes it's better to create the calculated measure first in the Calculation Designer and then reference it in the Value expression in the KPI Designer.

- **Goal**. This expression is the target against which the value expression is compared. The goal could be an expression or a fixed value. In the previous example, the goal is to achieve a one percent increase in gross profit margin from the previous year. The expression to define this goal uses a tuple to retrieve the Gross Margin Percent for the same time period in the previous year and multiple it by 1.01 to get the target value.

- **Status**. You use the status expression to quantify how close the value is to the goal and to express the result using a range from -1 to 1. The expression should return -1 when the goal is not achieved, 0 when the value is moving toward the goal but not meeting expectations, and 1 when the value is close to the goal, at the goal, or exceeding the goal. Business requirements determine where the boundaries between -1, 0, and 1 exist. For this example, let's say that a value of 1 is possible only when the value is within 85 percent of the goal. Otherwise, the status is 0, unless the value is below 50 percent of the goal, in which case the status returns -1. As part of the status definition, you also choose an image to associate with the status results. You can later create a dashboard constructed of KPIs (using Excel or PerformancePoint Services in SharePoint, for example) that tell you at a glance whether goals are being met.

- **Trend**. The trend expression compares a value in one time period to a previous period. It's up to you to determine whether that time period is the previous period, a year-over-year period, or a moving average type of trend. As with status, you define the expression to return a value between -1 and 1. A -1 value is a negative trend that is displayed as an arrow pointing downward, whereas a 1 value is a positive trend that displays an arrow pointing upward. In the previous example, the trend expression performs a simple comparison of the current period to the previous period.

Unlike the Calculation Designer, you can use a browser built into the KPI Designer to check the results of your KPI definition. You must first deploy your project to save the KPI definition, and then click the **Browser View** button (the tenth button on the KPI Designer toolbar). In the browser, you can define filters to observe the changes to the KPI results, as shown in Figure 78. To view status and trend values that depend on time comparisons, you must include a Date dimension in the filter.



| Dimension | Hierarchy | Operator | Filter Expression |
|---|---|---|---|
| Order Date | Quarter | Equal | { Q1 2008 } |
| <Select dimension> | | | |

| Display Structure | Value | Goal | Status | Trend | Weight |
|---|---|---|---|---|---|
| Gross Profit | 0.32% | 0.04 | ◆ | ↓ | |

*Figure 78: Key Performance Indicator Browser View*

***Tip: After changing a dimension filter in the browser view, click anywhere in the Display Structure area to apply the filter and update the KPI values.***

## Actions

Another way to enhance a cube is to define an action on the Actions tab of the Cube Designer. An **action** is a feature of Analysis Services that allows the user to initiate a process in the context of a current query, such as opening a webpage or launching a report. You can add any of three types of actions to a cube: a standard action, a drillthrough action, or a reporting action.

Regardless of its type, an action has a target, which is the object in the cube that the user clicks to trigger the action. A target can be a member of an attribute, hierarchy, level, or dimension; a cell, hierarchy, or level; or even the cube itself. When a user explores the cube, the client application displays available actions associated with a target. The steps to display or launch an action depend on the client application. For example, Excel displays available actions when you right-click a target, such as Mountain Bikes, as shown in Figure 79. In this case, the action is assigned a target type of attribute members and a target object of Product.Subcategory is displayed in Excel. In the submenu, click **Additional Actions** to view the available actions, such as “Search for Mountain Bikes” in this example.

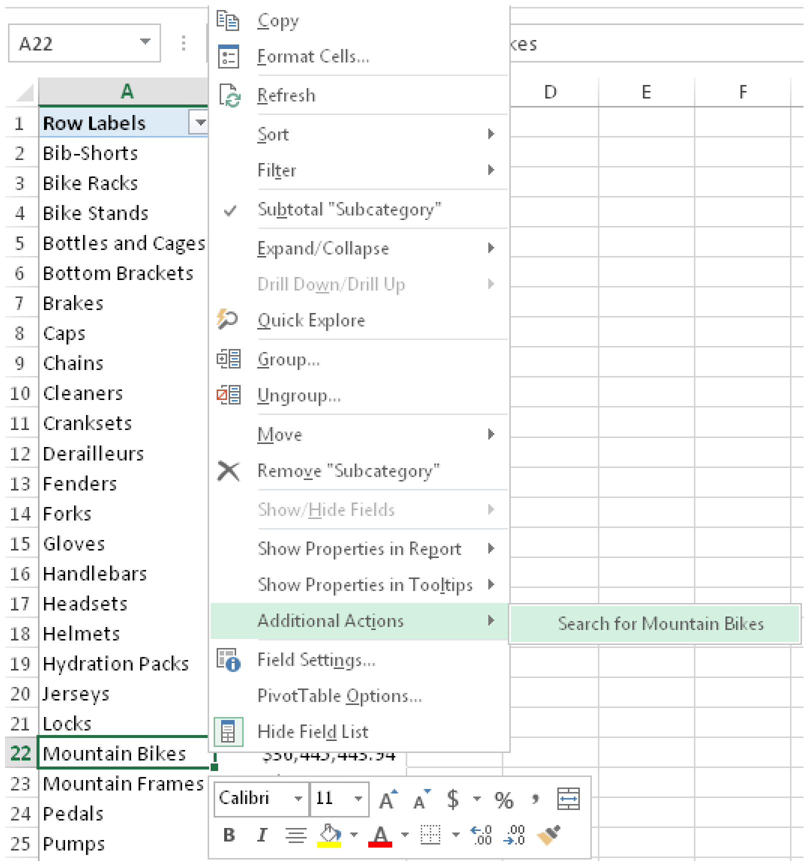


*Figure 79: Action Visible in Excel*

***Note: Before adding actions to your cube, be sure the client application supports this feature.***

## Standard Action

A standard action is an action that returns data or launches another application based on the selected target. When you define an action, you specify an MDX expression that resolves as a string that the action provides to another application for execution. You also assign one of the following action types to your action:

- **Dataset**. This type of action returns a multidimensional dataset (also known as a Cellset object) to the client application after resolving a string containing an MDX statement, which executes using an Analysis Services OLE DB provider, ADOMD.NET, or XML for Analysis.
- **Proprietary**. This type of action performs a task defined by the client application, which is also responsible for interpreting the string expression.
- **Rowset**. This type of action returns a rowset to the client application after resolving a string expression as a multidimensional or relational query. It differs from the dataset action type because it can use any OLE DB-compliant data source.
- **Statement**. This type of action executes an OLE DB command after resolving a string as an MDX statement, such as a CREATE SUBCUBE statement. The result returned by the action is the status of statement execution, success or failure.
- **URL**. This type of action opens a webpage in a browser window after resolving a string as a URL.

***Note: Most of these action types require the use of a custom application to interpret and execute the action and to use the results of execution in some way. If you are using Excel or PerformancePoint Services, only the URL action is supported.***

Because the URL action type is the most commonly used of the available action types, let's consider an example in which the action must open a webpage that displays Bing search results for a selected product subcategory. To create the action, click **New Action** (the fifth button in the toolbar) and assign a name. In this case, the target Type is Attribute Members and the Target Object is Product.Subcategory. For this action, you create an Action Expression using an MDX expression that concatenates static text and the name of the selected member like this:

```
"http://www.bing.com/search?q=" +
[Product].[Subcategory].CurrentMember.Name
```

You can also configure other settings for an action by expanding the **Additional Properties** section for the action:

- **Invocation**. You specify how the action occurs. If the user triggers the action, such as a URL action, this setting should be **Interactive**. You can also choose **Batch** to run the action as part of a batch operation or **On Open** to run it when the user opens the cube.

- **Application**. Here you type the name of the application to run the action as a recommendation to the client application from which the action is invoked. It is not a required property.

- **Description**. This property is also optional and is simply used as metadata to document the action.

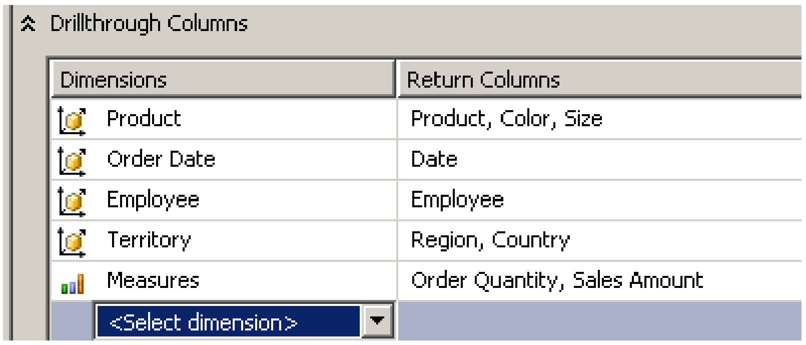
- **Caption**. The client application displays the caption that you provide for this property. You can use static text or dynamically generate the caption by using an MDX expression. For example, you could create a custom caption for the URL action to display a string containing the name of a selected attribute member such as “Mountain Bikes” by using the following expression:

```
"Search for " + [Product].[Subcategory].CurrentMember.Name
```

- **Caption is MDX**. The default value for this property is **False**. If you use an MDX expression for the Caption property, be sure to change this property to **True**.

## Drillthrough Action

You add a drillthrough action to allow users to see the underlying detail data associated with a selected cell. To create the action, click **New Drillthrough Action** (the sixth button in the toolbar), assign a name, and select a specific measure group or all measure groups as the action target. In the **Drillthrough Columns** section, you select one or more combinations of dimensions and attributes to return in the drillthrough results, as shown in Figure 80.


Drillthrough Columns

| Dimensions | Return Columns |
|---|---|
| Product | Product, Color, Size |
| Order Date | Date |
| Employee | Employee |
| Territory | Region, Country |
| Measures | Order Quantity, Sales Amount |
| <Select dimension> | |

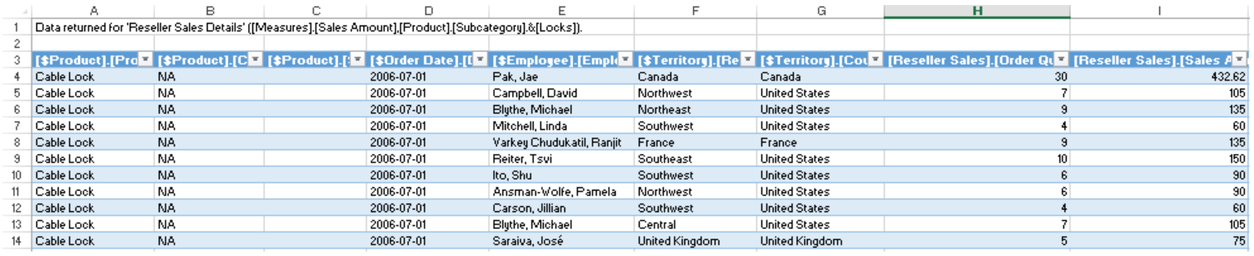
*Figure 80:  Drillthrough Columns Definition for Drillthrough Action*

You can configure additional properties like those available for a standard action, as well as the following properties specific to the drillthrough action:

- **Default**. Leave this value as **False**. It is available as a property to support backward compatibility when the client application executes a DRILLTHROUGH statement.

- **Maximum Rows**. You can specify a limit on the number of rows that Analysis Services returns when performing the drillthrough action.

In the client application, you invoke the drillthrough action from a cell containing a measure in the measure group specified as the action target. Figure 81 shows a portion of the data returned by a drillthrough action in Excel.



Data returned for 'Reseller Sales Details' ([Measures].[Sales Amount],[Product].[Subcategory].&[Locks]).

| [$Product].[Pro | [$Product].[C | [$Product].[ | [$Order Date].[ | [$Employee].[Empl | [$Territory].[Re | [$Territory].[Cou | [Reseller Sales].[Order Qu | [Reseller Sales].[Sales A |
|---|---|---|---|---|---|---|---|---|
| Cable Lock | NA | | 2006-07-01 | Pak, Jae | Canada | Canada | 30 | 432.62 |
| Cable Lock | NA | | 2006-07-01 | Campbell, David | Northwest | United States | 7 | 105 |
| Cable Lock | NA | | 2006-07-01 | Blythe, Michael | Northeast | United States | 9 | 135 |
| Cable Lock | NA | | 2006-07-01 | Mitchell, Linda | Southwest | United States | 4 | 60 |
| Cable Lock | NA | | 2006-07-01 | Varkey Chudukatil, Ranjit | France | France | 9 | 135 |
| Cable Lock | NA | | 2006-07-01 | Reiter, Tsvi | Southeast | United States | 10 | 150 |
| Cable Lock | NA | | 2006-07-01 | Ito, Shu | Southwest | United States | 6 | 90 |
| Cable Lock | NA | | 2006-07-01 | Ansman-Wolfe, Pamela | Northwest | United States | 6 | 90 |
| Cable Lock | NA | | 2006-07-01 | Carson, Jillian | Southwest | United States | 4 | 60 |
| Cable Lock | NA | | 2006-07-01 | Blythe, Michael | Central | United States | 7 | 105 |
| Cable Lock | NA | | 2006-07-01 | Saraiva, José | United Kingdom | United Kingdom | 5 | 75 |

*Figure 81: Drllthrough Results*

## Reporting Action

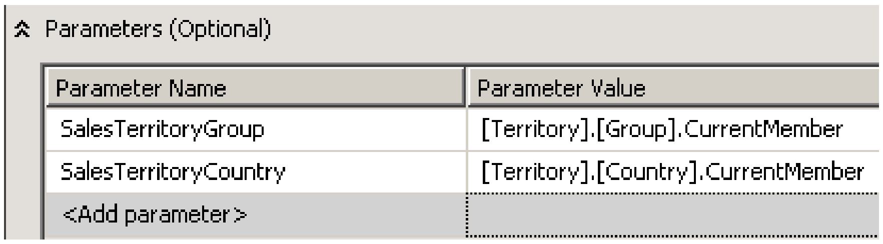
When a user launches a reporting action, the current context of the user's selection can pass to a report hosted in Reporting Services as one or more parameters. The reporting action requires you to specify a target type and target object just like a standard action. You also specify the following report server properties:

- **Server name**. Here you provide the URL for the report server's web service. This value is typically *http://<servername>>/reportserver*, where <*servername*> is the name of the computer hosting Reporting Services for a native mode installation, or *http://<servername>/<site>* for a SharePoint integrated mode installation, where <*servername*> is the SharePoint server name and <*site*> is the name of the site containing the document library for reports.

- **Report path**. The report path is the folder or path of folders containing the report and the name of the report to execute. For example, to launch a report called Total Sales by Territory in the Sales folder on a native-mode report server, you define the following report path: /Sales/Total Sales by Territory.

- **Report format**. You can launch the report in the default web browser by selecting the **HTML3** or **HTML5** options, or open the report in Adobe Reader or Excel by selecting the **PDF** or **Excel** options, respectively.

If the report contains parameters, you can provide static text or dynamic MDX expressions to configure parameter values for report execution. You type the parameter name as it appears in the report definition language (RDL) file and provide an expression that resolves to a value that is valid for that parameter, as shown in Figure 82.


Parameters (Optional)

| Parameter Name | Parameter Value |
|---|---|
| SalesTerritoryGroup | [Territory].[Group].CurrentMember |
| SalesTerritoryCountry | [Territory].[Country].CurrentMember |
| <Add parameter> | |

*Figure 82: Parameter Definition for Reporting Action*

## Writeback

Writeback is one of the less commonly used features of Analysis Services for two reasons. First, it requires a client application that supports it. Second, writeback is used by a relatively small number of users who need to do planning, budgeting, or forecasting, as compared to the number of users performing analysis only. In fact, the most common use of writeback in Analysis Services is in financial analytics, whether working on budgets by department or forecasting revenues and expenses. Before users commit to a final budget or forecast, they might go through several iterations of planning and can use writeback to support that process.

***Note: In this section, I review the concepts for cell writeback and dimension writeback. To learn more about writeback, see Building a Writeback Application with Analysis Services.***

### Cell Writeback

Data normally becomes part of a cube when events occur that we want to track, such as sales or general ledger entries. However, with writeback, users can update the cube with values that they anticipate might occur, and can use the aggregation capabilities of Analysis Services to review the impact of these updates. To facilitate this activity, a client application that supports writeback is required to update a cell. In other words, writeback activity requires an update to fact data. To add data to the cube to describe what you expect or would like to see happen, you can use writeback to update values in a measure group at the intersection of selected dimension members. Typically, data captured for the writeback process is at a high-level, such as a quarter, and the Analysis Services database must contain instructions for allocating to leaf-level members like days.

Analysis Services stores the writeback data in a separate ROLAP partition, regardless of the cube storage mode (as described in Chapter 4, "Developing cubes"). In fact, the writeback process updates a relational table and then performs an incremental process of the MOLAP partition to bring it up-to-date to match the source. Any subsequent queries to the cube return data from the MOLAP partition only. The cost of this incremental update is small compared to the impact of ROLAP queries.

To implement cell writeback, you create a writeback partition for a measure group, which can use MOLAP or ROLAP storage. As a user adds or changes values in the cube, the values stay in memory for the user's current session and no one else can see those values. That way, a user can test the impact of changes made before finalizing the input. When finished, a COMMIT command in the writeback application recalculates user input as weighted values to leaf-level cells if the input was at higher levels. After that, T-SQL Insert statements are executed to add corresponding records to the writeback partition. If the partition uses MOLAP storage, Analysis Services performs an incremental update (as described in Chapter 6, "Managing Analysis Services databases") to bring the cube up-to-date for faster performance.

***Note: You can use cell writeback in any edition of Analysis Services.***

## Dimension Writeback

You can implement dimension writeback when you want to give users the ability to add new members to the dimension without requiring them to add it to the source application. However, you must have a client tool that supports dimension writeback. You can also use security to control who can use this feature, as explained in Chapter 6.

Dimension writeback is helpful when a source application is not available to maintain the members, such as a Scenario dimension. As an example, you might have a dimension like this in the data warehouse for which you preload it with members but don't have a regular extract, transform, and load process to continually add members over time. In the AdventureWorksDW sample database, the DimScenario table has only three members to start: Actual, Budget, and Forecast.

To implement dimension writeback, you need to modify the dimension design. Specifically, you change the **WriteEnabled** property from its default value of **False** to **True**, and then deploy the change. If users are working with an application that supports dimension writeback, they can then add new members as needed. If you have multiple versions of budgets or forecasts that you want to keep separate during the planning process, you can also use dimension writeback to create a name used to track each version. For example, in a Scenario dimension, each member is either associated with actual data or a specific version of a budget or forecast, such as Budget v1 or Forecast v2.3.

When a user adds a new member with dimension writeback, Analysis Services updates the relational source immediately. For this reason, you can use dimension writeback only with a dimension that has a single table or an updateable view as its source. An incremental update (as described in Chapter 6) then gets the data from the relational source to update the Analysis Services database, and any subsequent queries to get dimension metadata include the new member. There is no separate storage of dimension members added in this way.

***Note: You can only use dimension writeback in the Enterprise or Business Intelligence editions of Analysis Services.***

# Chapter 6 Managing Analysis Services Databases

Our focus up to this point has been the development process. Periodically during development, you use the Deploy command in SSDT to deploy and process the Analysis Services database. When you do this, you are able to browse dimensions and launch Excel to browse the cube. However, you have yet to learn what it really means to deploy and process a database. In this chapter, I start with an overview of deployment and how to use the Deployment Wizard to move an Analysis Services database from development to production. Then I explain what happens during processing. Next, I show you how to configure security to enable users and administrators to access the database. Lastly, I review the alternatives you have for making copies of your database for disaster recovery and for scale-out.

## Deployment Options

There are two ways to deploy an Analysis Services database after completing development tasks:

- Deploy Command
- Deployment Wizard

### Deploy Command

In Chapter 3, "Developing dimensions," you learned how to use the Deploy command on the Build menu in SSDT. When you make changes to the Analysis Services database design, whether updating a measure group in a cube or changing an attribute in a dimension, you must deploy the changes made in the development environment so that the database on the server reflects those changes, which allows you to review the results. Now let's take a closer look at the three steps this command performs:

- Build
- Deploy
- Process

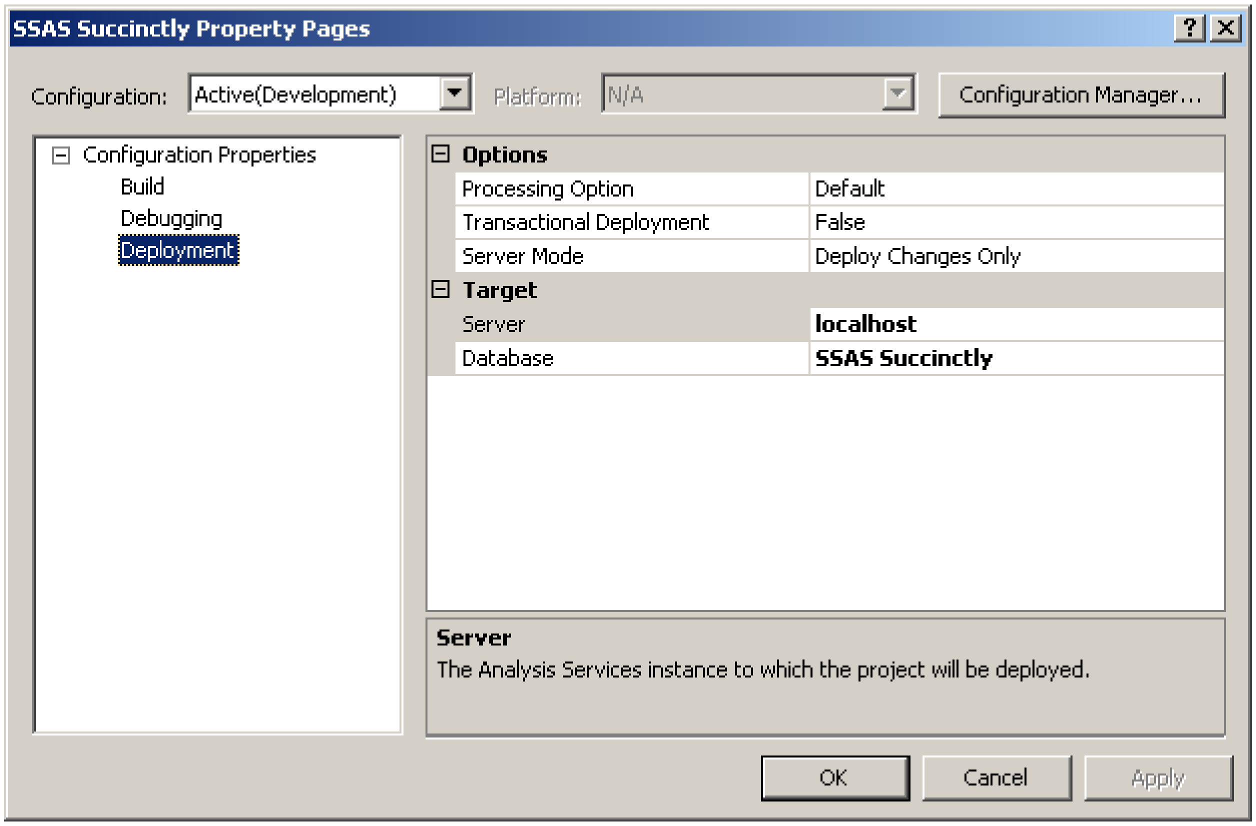
The **Build** step consolidates the database object definitions in the project from each of the following files in the project: DS, DSV, CUBE, DIM, and ROLE. The result of the consolidation is a single ASDATABASE file that contains the XML definitions for each of the database objects.

The **Deploy** step copies the ASDATABASE file to the Data folder for the Analysis Services server. By default, this folder is located at /Program Files/Microsoft SQL Server/MSAS11.MSSQLSERVER/OLAP. The individual object definitions are then read from the file and decomposed into separate files again, matching the file structures in the SSDT project.

The operations performed in the **Process** step depend on the options defined on the Deployment page of the Project Properties dialog box. Here you can specify settings to determine whether processing occurs, whether deployment operations are transactional, and which objects to deploy.

You can choose from the following settings when configuring the **Processing Option** in Configuration Properties, as shown in Figure 83:

- **Default**. Analysis Services determines the type of processing to perform based on the changes it detects when you use the Deploy command.
- **Do Not Process**. Choose this option if you prefer to control the processing operation manually as a separate step following deployment.
- **Full**. Use this option if you want Analysis Services to fully process each object after you use the Deploy command.



| | |
|---|---|
| **Options** | |
| Processing Option | Default |
| Transactional Deployment | False |
| Server Mode | Deploy Changes Only |
| **Target** | |
| Server | **localhost** |
| Database | **SSAS Succinctly** |

**Server**
The Analysis Services instance to which the project will be deployed.

*Figure 83: Deployment Properties*

Another option in the Project Properties is the Transactional Deployment setting. By default, this setting is False, but you can change it to True if you prefer. When you use transactional deployment, the deployment and processing of the database objects must both succeed to persist the changes on the server. Otherwise, if either step fails, Analysis Services returns the database on the server to the state it was in prior to deployment of the changes.

The final setting affecting deployment is Server Mode, which has the following options:

- **Deploy Changes Only**. This is the default setting. Analysis Services deploys only the objects that do not exist or no longer match objects on the server. This is also the faster setting of the two choices.

- **Deploy All**. When you use this setting, Analysis Services copies all database objects to the server, replacing any existing objects if applicable.

## Deployment Wizard

As you now know, the deployment process places a new copy of changed files on the server, which the server uses to retrieve the current definition of database objects during processing. Although it's possible to use XMLA scripts to modify database objects, it's considered best practice to use SSDT to make design changes so that you can save your work in a source control system. However, on an ongoing basis, you might manually update Roles on the server to manage security or you might use scripts or automated processes to update the database object files or to add new partitions to hold newer data. In those cases, any partition or role definitions in SSDT that you deploy to the server would wipe out anything you update on the server directly by using XMLA scripts.

To better manage partitioning or roles after a database has been placed in production, or when you as a developer lack the permissions to deploy a database directly to production, a good alternative to use is the Analysis Services Deployment Wizard. Before launching the wizard, use the Build command in SSDT to generate the ASDATABASE file. Launch the wizard which is accessible by opening the Program Group menu for Microsoft SQL Server and then navigating to the Analysis Services folder. The wizard walks you through the following pages (after the Welcome page if it is still enabled):

- **Specify Source Analysis Services Database**. Here you must provide the path and file name for the ASDATABASE file, or click the ellipsis button to navigate through the file system to locate and open this file. The file is located by default in the bin folder of your project.

- **Installation Target**. On this page, you must specify the target server and target database names for your project on the Installation Target page.

- **Specify Options for Partitions and Roles**. Your choices here affect what happens to existing partitions and roles. By default, existing partitions will be replaced by whatever partition definition happens to be in your development files. If you have partitions on the server that you want to keep, you need to choose the replace option for partitions. In that case, any partitions in your database for measure groups that already exist are ignored and only new measure groups and their partitions are deployed to the server. On the other hand, the default for roles is to deploy new roles and keep the roles already on the server intact. Your other options would be to either replace all roles on the server with the roles that have been defined in the project, or to ignore anything in the project and keep the roles on the server intact.

- **Specify Configuration Properties**. You can configure the deployment process to keep any existing configuration and optimization settings already stored on the server. In addition, you can change the connection string to the data source, impersonation information for processing, locations for error files, the report location for reporting actions, and storage locations for the database cube, measure groups, or partitions.

- **Select Processing Options**. Here you have the option to set values for the project property settings related to processing: Processing Method and Transactional Deployment. If your cube is configured to support writeback, you have an additional property to set that controls whether Analysis Services should use an existing table for writeback or create a new table.

- **Confirm Deployment**. You can continue through the wizard without creating a deployment script if you prefer to have the wizard invoke the deployment process. On the other hand, if you choose to create a deployment script, you must specify a location for storing the XMLA file that the wizard generates. You can later open this file in SSMS and execute it to perform deployment.

## Processing Strategies

When you first deploy an Analysis Services database, it is in an unprocessed state and empty. That means no one can query the database. You need to perform a full process of the database to load the dimensions and cube partitions with data. After the initial process, you have the following options for keeping data up-to-date in the database:

- Full process
- Process data and process index
- Process update
- Process add

### Full Process

If the amount of time required to do a full database process is small enough, you can choose to do a full database process on a nightly basis. You can schedule the full process as a SQL Server Agent job or include it as an Analysis Services Processing Task in an Integration Services package, as shown in Figure 84. However, for large cubes that take longer to process, you might need to implement other strategies. Each time you do a full process, the existing contents are removed from the database and the data is completely reloaded in all dimensions and cube partitions. You can choose to do a full process at the database level, cube level, measure group level, or dimension level. To configure the Analysis Service Processing Task, select the target object(s) for the process, and select **Process Full** in the **Process Options** drop-down list.

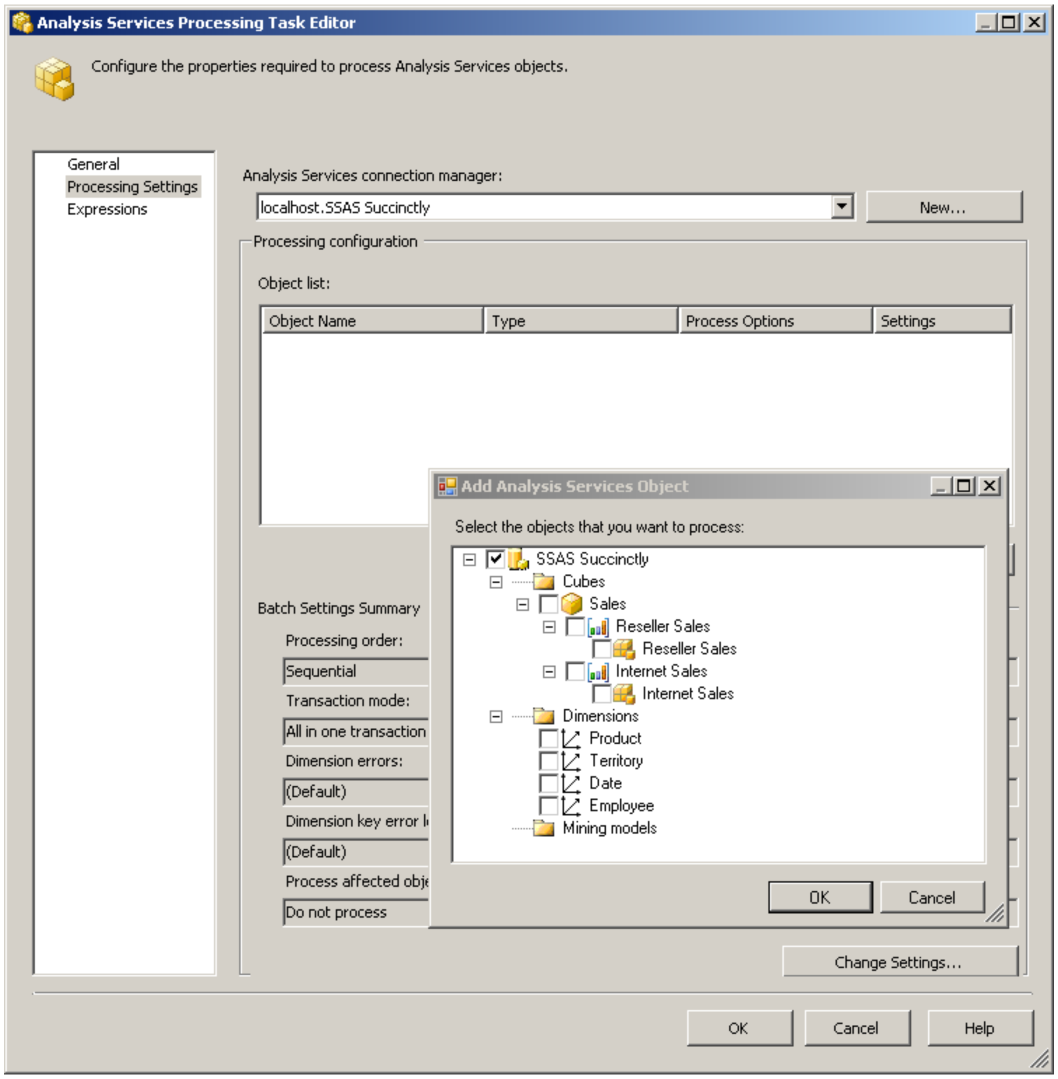


*Figure 84: Selection of Objects to Process*

A similar user interface appears when you right-click a database, cube, measure group, or dimension in SSMS and select the **Process** command on the submenu. However, in SSMS, you cannot select multiple objects to process in the same dialog box. On the other hand, if you open the **Process** dialog box, you can click **Script** in the dialog box toolbar to generate a script like this:

```
<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
  <Parallel>
    <Process xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ddl2="http://schemas.microsoft.com/analysisservices/2003/engine/2"
xmlns:ddl2_2="http://schemas.microsoft.com/analysisservices/2003/engine/2/2
"
xmlns:ddl100_100="http://schemas.microsoft.com/analysisservices/2008/engine
/100/100"
xmlns:ddl200="http://schemas.microsoft.com/analysisservices/2010/engine/200
```

```
"
xmlns:ddl200_200="http://schemas.microsoft.com/analysisservices/2010/engine
/200/200"
xmlns:ddl300="http://schemas.microsoft.com/analysisservices/2011/engine/300
"
xmlns:ddl300_300="http://schemas.microsoft.com/analysisservices/2011/engine
/300/300">
      <Object>
        <DatabaseID>SSAS Succinctly</DatabaseID>
        <CubeID>Sales</CubeID>
      </Object>
      <Type>ProcessFull</Type>
      <WriteBackTableCreation>UseExisting</WriteBackTableCreation>
    </Process>
  </Parallel>
</Batch>
```

You can repeat this procedure to generate a script for each object to process. You can then combine the Object elements from each script below the Process node in the first XMLA script you create. That way, you can process multiple objects in parallel. If you prefer to process objects sequentially (which might be necessary if your server memory is constrained), you can remove the Parallel element from the script.

The Process dialog box, in both the Analysis Services Processing Task and SSMS, includes a **Batch Settings Summary** section. Click **Change Settings** in this section to update the following settings:

- **Processing Order**. The default processing order is **Parallel**. You can let the server decide how many processing tasks to execute in parallel or set a specific maximum number of parallel tasks. As an alternative, you can specify sequential processing.

- **Transaction Mode**. If you process tasks sequentially, you can specify whether to treat each task as a single transaction or as separate transactions.

- **Dimension Errors**. You can keep the default error configuration to automatically fail processing if a problem occurs during dimension processing, or define a custom error configuration as described in Processing Options and Settings.

- **Dimension Key Error Log Path**. You can specify a path in which Analysis Services will store a log file containing dimension key errors if you set up a custom error configuration.

- **Process Affected Objects**. You can instruct Analysis Services to process all objects that have a dependency on the selected object.

***Note: If you perform a full process on a dimension, then you MUST do a full process on any associated cube.***

One way to optimize processing is to create partitions in a cube as described in Chapter 4, "Developing cubes." A common partitioning strategy is to create separate partitions for separate time periods. Consequently, it becomes necessary only to process the most current partition because data in older partitions rarely changes. Therefore, you can limit the time necessary for processing a cube by running a full process only for the current partition. Analysis Services removes the data in that partition only, reloads data from the source, and rebuilds aggregations. If the current partition is relatively small compared to the overall cube size, the full process on the partition will be faster and more efficient than processing all partitions. All other existing partitions remain unchanged.

### Process Data and Process Index

When working with a large cube or dimension, you might find it more effective to replace a Process Full operation with Process Data and Process Index operations. By splitting up these two operations that are included as part of Process Full, you can enable Analysis Services to focus resources on one operation at a time and thereby get better performance. However, this is typically necessary only when working with large data volumes.

### Process Update

Another option that takes less time than the Process Full operation is to only add new data or update changed data to the database. This option is applicable only to a dimension. The Process Update operation loads all new dimension members and makes changes as necessary to existing dimension members.

Performing a Process Update does not require you to execute a Process Full operation on a cube as is necessary after executing Process Full on a dimension. Inside a cube, there are internal identifiers for the dimension members that are created when storing data in the cube partitions. When you run Full Process on a dimension, even when loading in the same data, there is no guarantee that each dimension member gets the same internal identifier, so the cube becomes invalid. With Process Update, the identifiers remain the same. The only potential problem is that if you move a member from one parent in a hierarchy to another parent, and if flexible attribute relationships are defined, any aggregations for that hierarchy are invalidated. The cube can be queried, but queries can potentially run more slowly until aggregations are rebuilt with a Process Index or Process Default operation on the cube.

### Process Add

If dimension data that is already loaded doesn't change and there are only new dimension records to add periodically, then you can use the Process Add operation to insert the new records. This operation runs faster than Process Update and has no impact on the cube. It is not necessary to rebuild aggregations nor process the cube when adding new dimension members. On the other hand, because Process Add requires some additional configuration, it's not a particularly straightforward option. Greg Galloway, a SQL Server MVP and BI architect at Artis Consulting, has blogged about Process Add Examples with explanations that you can adapt to suit your needs.

If you are using Standard edition, you can maintain a maximum of three partitions per measure group, but you can still manage new partition data efficiently by using Process Add to insert that data into an existing partition. (Incidentally, this shows up as Process Incremental when you're using the graphical interface to configure processing.) This approach is likely to run much faster than running Process Full on the partition. However, it's important not to let this become the only form of processing you perform, because some fragmentation can occur when you do this often. Consider running Process Add nightly and then running Process Full on a weekly or monthly basis. Aggregations are updated with Process Add, so there is no need to run a separate process afterwards.

# Security

As you build and deploy the database throughout the development process, its contents—the cube and dimensions—are secure by default. No one besides the database developer is able to query the cube. To make the database accessible to user queries, there are a few extra tasks to perform. In addition, you might need to configure administrative security for users who need to perform processing tasks. Lastly, the Analysis Services service account requires specific permissions to support processing and, when applicable, query logging and writeback.

## User Security

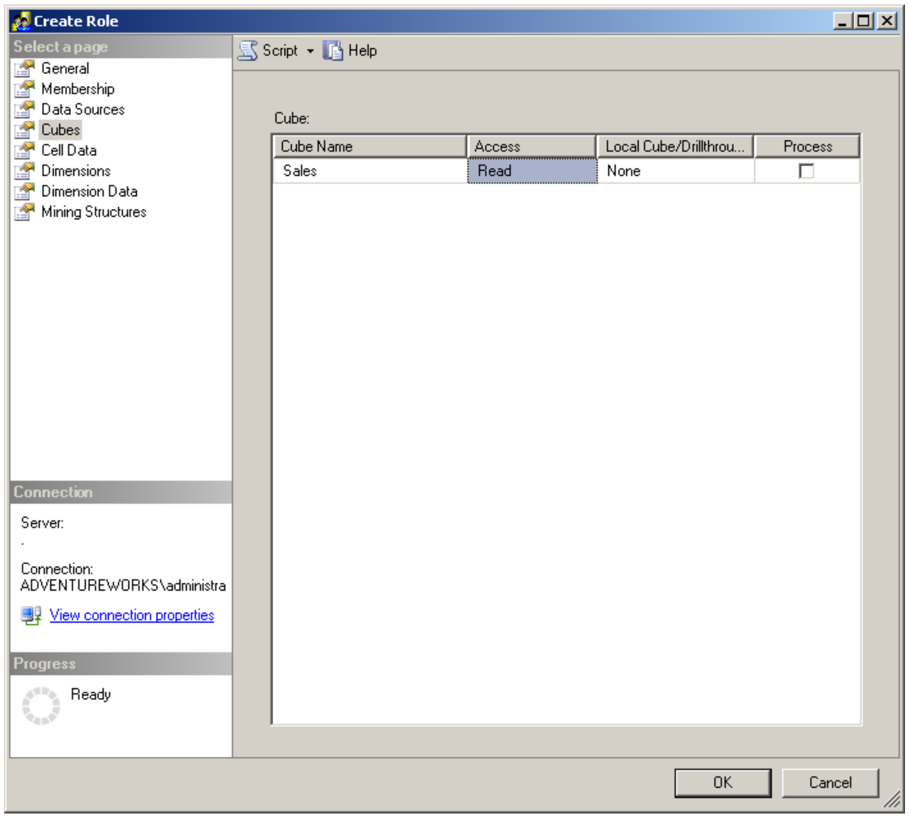
You must grant users access to Analysis Services at the cube level in each database. Before you can do this, each user must have a Windows login and can optionally be assigned to a Windows group in your network's Active Directory. Analysis Services uses role-based security, which has a dependency on Windows authentication. At minimum, you create a role to which you assign Windows logins, or better yet, Windows groups, and grant the role permission to access a cube. For more precise security, you need to take additional steps. For example, you can configure security at the dimension level, choosing the members you want users to see and excluding all others. You can also get more granular with security by controlling which cells a user can or cannot see.

### Role Membership

To create a role, either in SSDT or in SSMS after opening a database folder, right-click the **Roles** folder, select **New Role**, and type in a role name. There is no need to assign database permissions to users who only need the ability to query the cube. On the **Membership** page of the **Create Role** dialog box, click **Add** and retrieve user logins or Windows group names from Active Directory.

### Cube Security

After creating roles and assigning users or groups to them, you must explicitly grant permissions to authorize what users can see or do. At minimum, you need to assign the role **Read** access to each cube on the **Cubes** page of the **Create Role** dialog box in SSMS, as shown in Figure 85, or on the **Cubes** tab of the Role Designer in SSDT. You can also optionally specify whether the role will have permission to use drillthrough functionality or create a local cube using a client tool.



Server:
.

Connection:
ADVENTUREWORKS\administra

Cube:

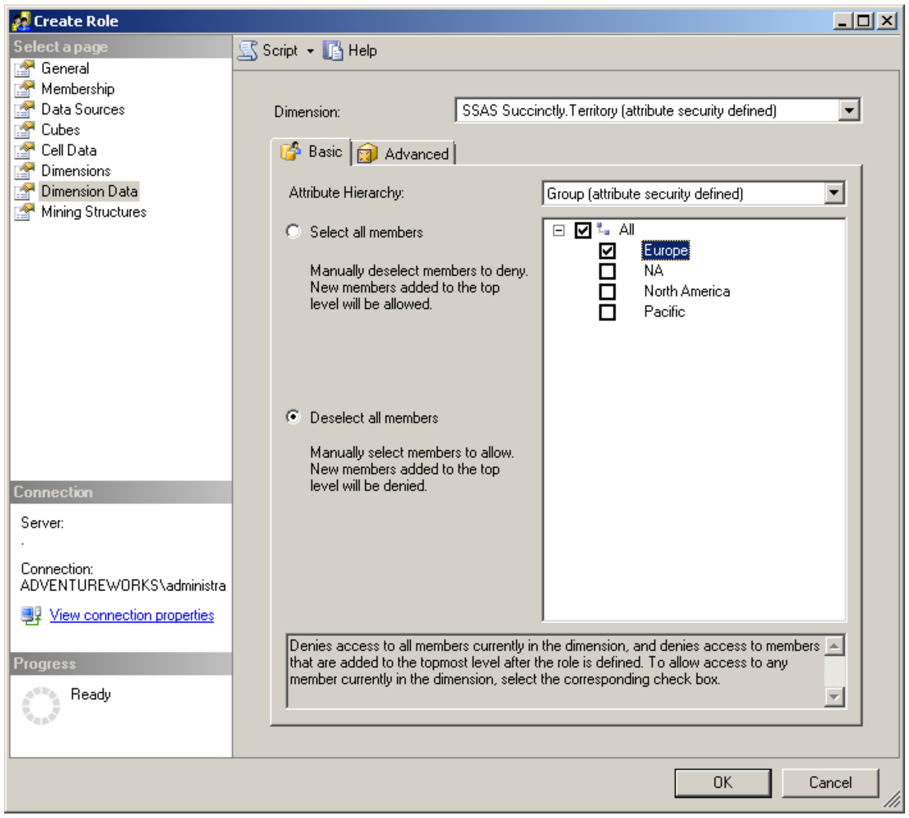
| Cube Name | Access | Local Cube/Drillthrou... | Process |
|---|---|---|---|
| Sales | Read | None | ☐ |

*Figure 85: Assignment of Read Access to Cube*

## Dimension Security

With dimension security, you can create sets of dimension members that are either allowed or denied for any or all attribute hierarchies in a dimension. For example, if you create a set of allowed members, the user can only see the members assigned to that set. Conversely, if you configure a denied set, the user can see all members in the attribute hierarchy except the members in that set. In general, dimension security restricts access to members of a specific attribute hierarchy.

Let’s say that you want to allow a role to see only Europe in the Group attribute hierarchy in the Territory dimension. On the **Dimension Data** page, select the **Territory** dimension in the drop-down list at the top of the page, and then select **Group** in the **Attribute Hierarchy** drop-down list. Select **Deselect All Members** to clear the selection of members, and then select the **Europe** check box, as shown in Figure 86.



Dimension: SSAS Succinctly.Territory (attribute security defined)

Attribute Hierarchy: Group (attribute security defined)

( ) Select all members

Manually deselect members to deny. New members added to the top level will be allowed.

(•) Deselect all members

Manually select members to allow. New members added to the top level will be denied.

- [✓] All
  - [✓] Europe
  - [ ] NA
  - [ ] North America
  - [ ] Pacific

Denies access to all members currently in the dimension, and denies access to members that are added to the topmost level after the role is defined. To allow access to any member currently in the dimension, select the corresponding check box.

Server:
.

Connection:
ADVENTUREWORKS\administra

View connection properties

Ready

*Figure 86: Dimension Security*

***Note: On the Advanced tab for dimension data, you can use MDX expressions to define allowed or denied sets rather than explicitly selecting members for inclusion on the Basic tab.***

Dimension security affects what the user sees in the metadata for the cube. Figure 87 shows the metadata tree from SSMS for a user assigned to a role that has only Europe as an allowed member in the Group attribute hierarchy. As you can see, none of the other members of the attribute hierarchy are visible: NA, North America, and Pacific. The user simply cannot see that these other members exist in the dimension. Notice also in the user-defined Territories hierarchy that Europe is again the only member visible at the Group level, but the children of Europe—France, Germany, and United Kingdom—are also visible even though the dimension security did not explicitly exclude those members. Furthermore, the Country attribute hierarchy is also restricted to only those countries that are children of Europe in the user-defined Territories hierarchy. The attribute relationship definition for the Territory dimension controls the security across all hierarchies in the dimension based upon the selection of allowed or denied members defined in dimension security.

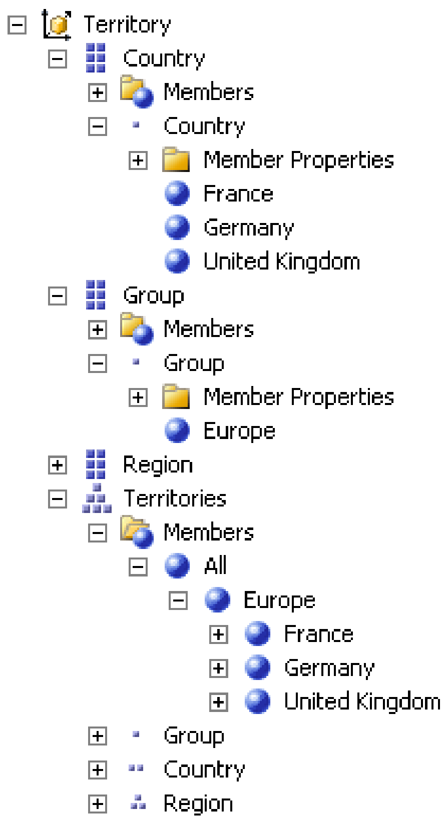
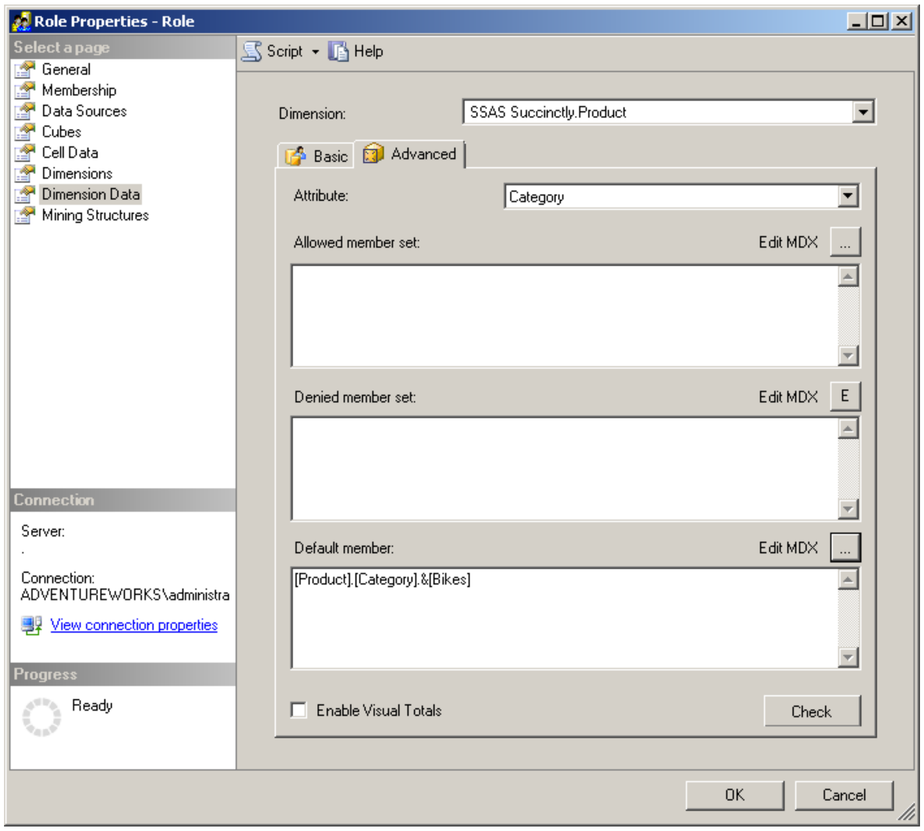


*Figure 87: Visible Members Across Hierarchies*

Another option available when configuring dimension security is specification of a default member. By defining a default member, you are adding an automatic filter to the dimension for that member. For example, let's say that you have department managers who want to see sales for their respective departments. You can create a role for each department and then configure a default member for each role—one for the Bikes manager, one for the Accessories manager, and so on. That way, every time managers run a query, they don't have to explicitly ask for their department's category because Analysis Services automatically applies the filter for their respective roles. However, this is not the same as restricting access to dimension members. If managers asked for all categories in a query, they could get that information as long as no dimension security restricted that information. To add a default member, select the applicable dimension and attribute hierarchy on the **Basic** tab of the **Dimension Data** page, open the **Advanced** tab, and then provide the member-unique name, as shown in Figure 88.



Role Properties - Role

Select a page: General, Membership, Data Sources, Cubes, Cell Data, Dimensions, Dimension Data, Mining Structures

Connection

Server: .

Connection: ADVENTUREWORKS\administra

View connection properties

Progress: Ready

Dimension: SSAS Succinctly.Product

Basic | Advanced

Attribute: Category

Allowed member set:

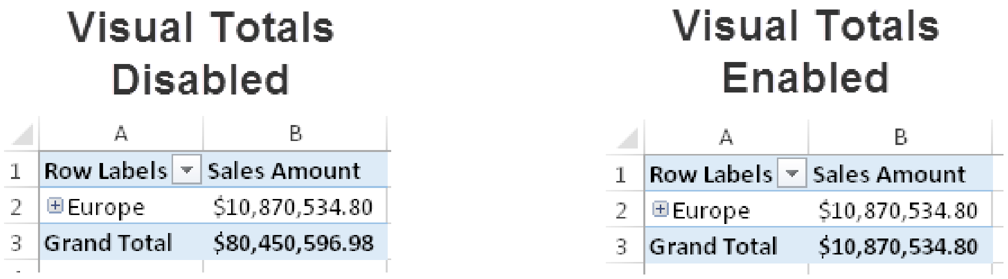
Denied member set:

Default member: [Product].[Category].&[Bikes]

☐ Enable Visual Totals

*Figure 88: Default Member Configuration*

You also have the ability to control whether the grand total for an attribute displays only the aggregation of the visible members or the aggregation of all members in the hierarchy. This feature is known as Visual Totals and is configurable on the **Advanced** tab of the **Dimension Data** page. By default, the Enable Visual Totals check box is not selected. In the case when dimension security restricts a user to Europe only, the grand total for the hierarchy displays the total for all groups, as shown on the left side of Figure 89. However, when the Enable Visual Totals check box is selected, the grand total is adjusted to show only the aggregation of the visible members, as shown on the right side of Figure 89.



Visual Totals Disabled

| | A | B |
|---|---|---|
| 1 | Row Labels | Sales Amount |
| 2 | ⊞Europe | $10,870,534.80 |
| 3 | Grand Total | $80,450,596.98 |

Visual Totals Enabled

| | A | B |
|---|---|---|
| 1 | Row Labels | Sales Amount |
| 2 | ⊞Europe | $10,870,534.80 |
| 3 | Grand Total | $10,870,534.80 |

*Figure 89: Effect of Visual Totals Feature*

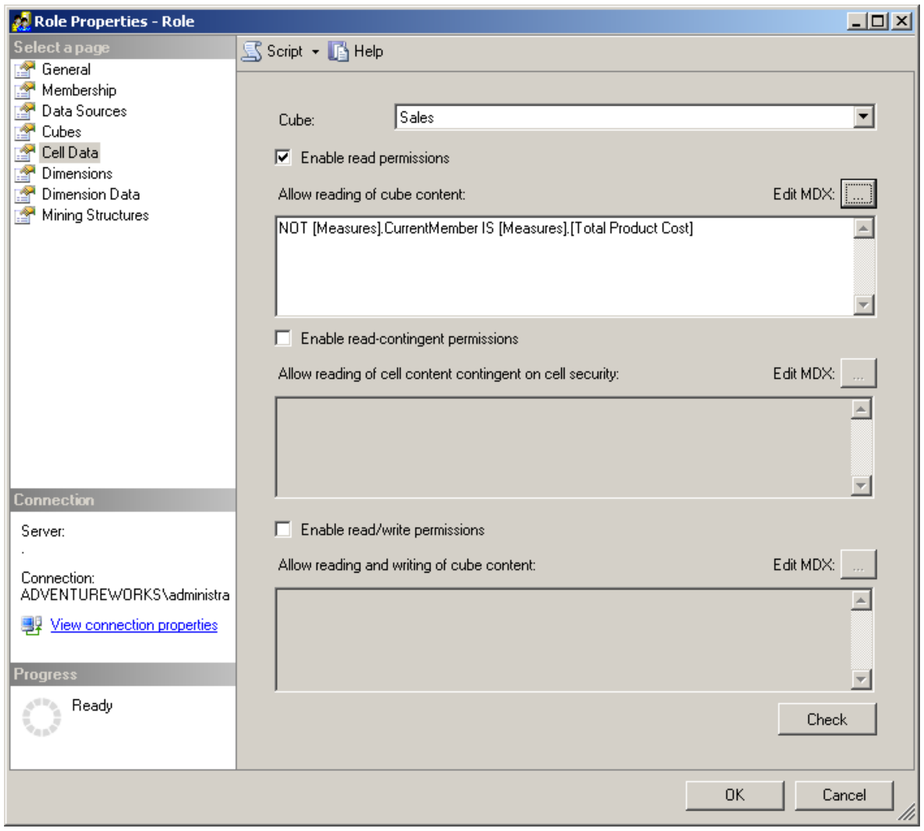
## Cell Security

Another way to manage security is to apply cell security. This will control whether the user sees a numeric value for certain intersections of measures and dimensions. You configure cell security on the **Cell Data** page and select a cube, and then you select one of the following permissions options:

- **Enable Read Permissions**. You provide an expression that evaluates as True or False to instruct Analysis Services whether to allow the user to see a value, as shown in Figure 90. In this case, the expression checks if the current member of the Measures dimension is Total Product Cost and displays a value only if the result is False. Therefore, only the Total Product Cost measure will not be visible to the user. Instead, a placeholder token is displayed to indicate the value is not available.

- **Enable Read-Contingent Permissions**. When you select this option, you also use an expression that returns True or False. Typically for read-contingent permissions, you reference a measure that is a calculation based on other measures. If the user is able to view all the other measures, based on Read permissions, the user can also view the calculation. For example, because Gross Profit Margin depends on Sales Amount and Total Product Cost, if Read-Contingent permission is assigned to Gross Profit Margin, the user can only view that value if the user has Read permission on Sales Amount and Total Product Cost.

- **Enable Read/Write Permissions**. If the cube is enabled for writeback, you can use this option to control which cells the user can update. Again, you use an expression that returns True or False.



Cube: Sales

[✓] Enable read permissions

Allow reading of cube content: Edit MDX:

```
NOT [Measures].CurrentMember IS [Measures].[Total Product Cost]
```

[ ] Enable read-contingent permissions

Allow reading of cell content contingent on cell security: Edit MDX:

[ ] Enable read/write permissions

Allow reading and writing of cube content: Edit MDX:

Server:
.

Connection:
ADVENTUREWORKS\administra

View connection properties

*Figure 90: Cell Security*

***Tip: If you have multiple measures to exclude from view or multiple conditions, you use the OR and AND operators to connect the expressions together.***

## Administrator Security

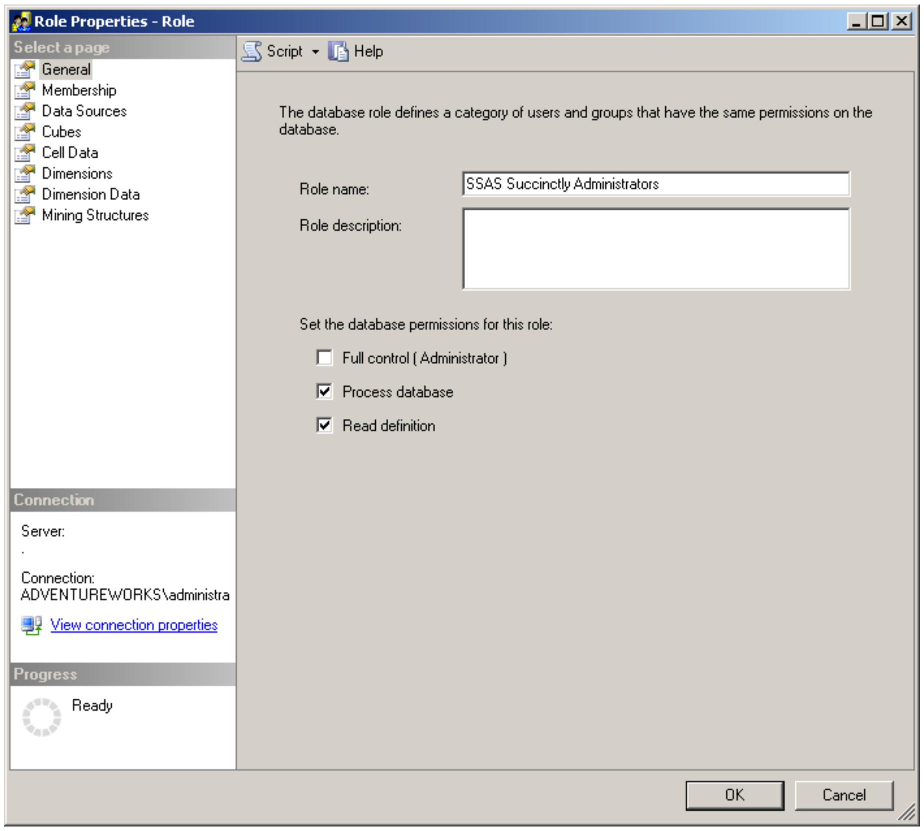
Besides user security, you might also need to configure administrator security. There are two levels of security for administrators: server and database.

### Server Administrator Security

In SSMS, right-click the Analysis Services server node in **Object Explorer** to access the server properties. In the **Properties** dialog box, the security page allows you to add Windows groups or users as server administrators. At the server level, these users can configure security options and access any database deployed to the server.

***Note: Built-in administrators on the server automatically have server-wide permissions, even without explicit assignment to the server administrator.***

You can also create database-level administrators. To do this, open a database and add a new role for administrators, and then define database permissions, as shown in Figure 91. For example, if you want a set of users to have the ability to process a database or read the XMLA definition of the database, you can create a role for those users and then grant those permissions at the database level. Having database permissions does not automatically grant access to the database contents. You must also grant cube access at minimum and define dimension and cell security where applicable.



The database role defines a category of users and groups that have the same permissions on the database.

Role name: SSAS Succinctly Administrators

Role description:

Set the database permissions for this role:

- [ ] Full control ( Administrator )
- [x] Process database
- [x] Read definition

Server:
.

Connection:
ADVENTUREWORKS\administra

Ready

*Figure 91: Database Permissions for Administrators*

## Service Account Security

When Analysis Services is installed, a service account is defined to run the windows service. When you define the service account, you can select a local user account, a domain user account, or a built-in system account. The best practice is to use a domain user account with low privileges.

Regardless of which type of account you use, you must make sure you grant the account permissions to access the data source for your Analysis Services database. If you use MOLAP or HOLAP storage for your cube, the service account requires only Read permission on the data source. However, if you use ROLAP storage, you must configure Read permission at minimum. If the ROLAP partition also includes aggregations, you must also configure Write permission.

When you install Analysis Services, the correct file system permissions should be set automatically. If the service account is changed without using the proper method, it's possible that the new account will lack the necessary permissions. In other cases, when you enable query logging or writeback, there is no automatic assignment of permissions, so you must manually update the applicable databases. In particular, the service account needs the following permissions:

- **Backup folder**. The service account must have Full Control permission on the Program Files\Microsoft SQL Server\MASA11.MSSQLServer\OLAP\Backup folder in order to save backup files.

- **Query Log Database**. If you enable the query log in preparation for usage-based optimization (described in Chapter 4, "Developing cubes"), the service account must be a member of the db_owner database role for the database in which the query log is to be created.

- **Writeback Database**. The service account also needs to be a member of the db_owner database role for the database storing the writeback table if you have writeback enabled on a partition in your cube.

***Note: To make sure all permissions are correct for the service account on the file system, be sure to use SQL Server Configuration Manger if you need to change the service account or password for local or domain user accounts.***

# Database Copies

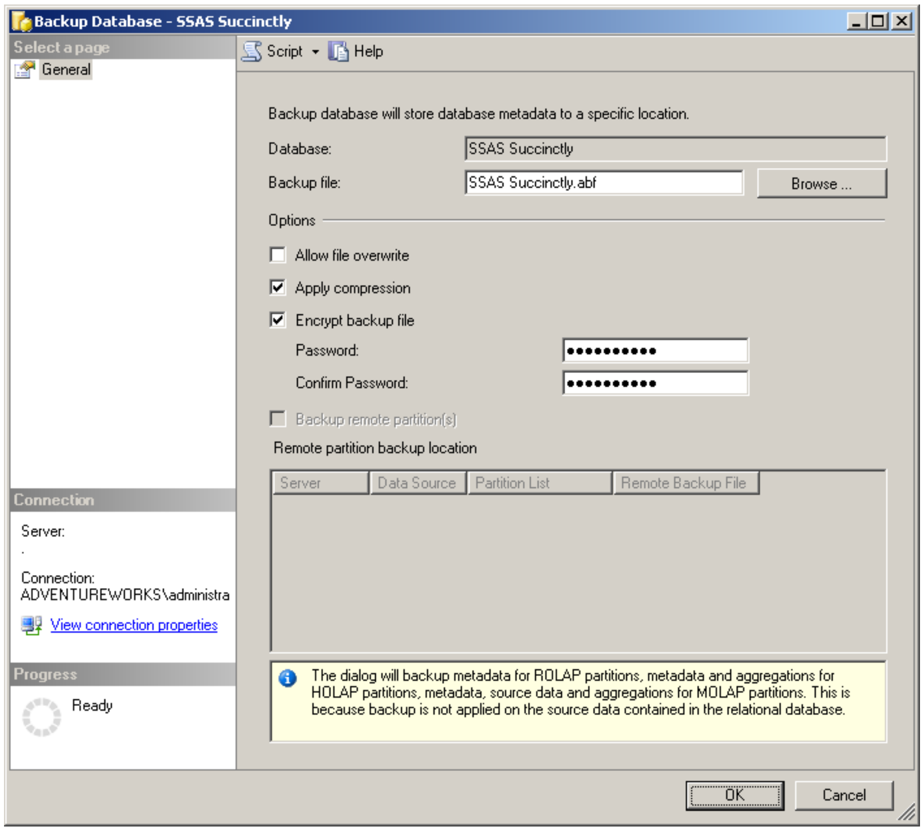
After putting an Analysis Services database into production, you should make a backup for disaster recovery. Backups are also a useful way to move a database from one server to another. For example, you might want to process a database on one server and then make it available on another server for querying, or you might want to place a copy on multiple servers to spread the query load across multiple servers. Whatever the reason, there is more than one way to copy and move a database.

## Backup and Restore

Backup and restore is a simple method to use. An advantage of this approach is that you do not have to add yet another operation to your normal routine, because you should already perform a backup each time you process the database. If you have a large database, you can get your database back into working order after a failure much more quickly from a backup than by performing a full process after redeploying the Analysis Services project files. If you are distributing a single Analysis Services database to multiple servers, you can create one backup, make as many copies as you need, distribute each copy to a separate server, and then run the restore operation.

***Note: Users cannot query the database during the restore process.***

To perform a backup manually, right-click the database in SSMS and select the **Back Up** command. In the dialog box, specify a name for the backup file and select options to control whether Analysis Services can overwrite an existing backup file, compress the file, or encrypt the backup with a password, as shown in Figure 92. As another option, you can click the **Script** button to generate an XMLA script. You can execute the script in an SSIS package by using an Execute DDL task, or as a SQL Server Agent job by using a SQL Server Analysis Services Command step.



Backup database will store database metadata to a specific location.

Database: SSAS Succinctly

Backup file: SSAS Succinctly.abf

Options

☐ Allow file overwrite

☑ Apply compression

☑ Encrypt backup file

Password: ●●●●●●●●●●●

Confirm Password: ●●●●●●●●●●●

☐ Backup remote partition(s)

Remote partition backup location

| Server | Data Source | Partition List | Remote Backup File |
|---|---|---|---|

The dialog will backup metadata for ROLAP partitions, metadata and aggregations for HOLAP partitions, metadata, source data and aggregations for MOLAP partitions. This is because backup is not applied on the source data contained in the relational database.

Server:
.

Connection:
ADVENTUREWORKS\administra

View connection properties

Ready

*Figure 92: Database Backup*

There is no limit to the size of a database you can back up. The actual contents of the backup depend on the storage mode you use for the cube partitions:

- MOLAP stores data, aggregations, and database metadata.
- HOLAP stores aggregations and metadata only.
- ROLAP stores metadata only.

## Synchronization

To use synchronization, you have one Analysis Services server to host the source database and a second Analysis Services server to host a synchronized copy. Right-click the database on the first server and run the **Synchronize** command to launch the process. Analysis Services compares the two databases to determine if there are any differences between them. If the databases are not identical, Analysis Services uses synchronization to update the second server and only sends the changes in compressed form. The number of changes affects performance. On the other hand, users can query a database during synchronization, unlike during the restore process. That said, when synchronization is complete, the second server switches out the older version of the database with the updated version, which doesn't take long, but during that period no queries are permitted. As with backups, you can script the synchronization process to automate it using SSIS or SQL Server Agent.

## Detach and Attach

With this method, you right-click a database on one Analysis Services server and select the **Detach** command. You can then move the associated database files to another server where you reverse the process. You right-click the Analysis Services server node, and select the **Attach** command. This method is a very fast way to make a database available on a second server. You can also put the database on a logical SAN in Read Only Mode and then have multiple servers attach the same database to create a scale-out solution.

# Chapter 7  Using Client Tools

The whole point of creating an Analysis Services database is to allow users to query the cube to quickly get answers to business questions. In this chapter, I introduce you to the tools in the Microsoft Business Intelligence stack and show you how to connect to Analysis Services using each of these tools. I also describe the key classes in the ADOMD.NET library that you can use to create your own client tool.

## Tools in the Microsoft Business Intelligence Stack

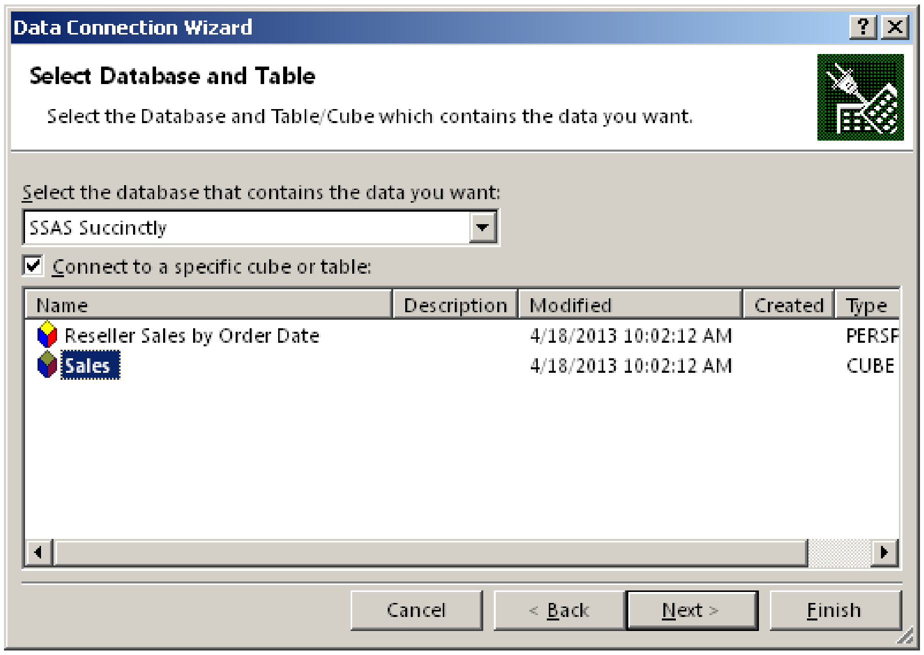
You can use a variety of tools in the Microsoft Business Intelligence stack to query an Analysis Services database. Some tools are intended primarily for reporting the data from a cube with minimal interaction from the user, whereas other tools are intended primarily for exploring the cube. In this section, I explain the basics for connecting the following tools to a cube and displaying its data:

- Microsoft Excel
- Microsoft SQL Server Reporting Services
- Microsoft SharePoint Server

### Microsoft Excel

Excel is a popular tool among business analysts who are often already familiar with its features. After connecting to a cube, pivot tables and pivot charts facilitate exploration of the data in a cube without requiring knowledge of the MDX query language.

To connect to a cube in an open Excel workbook, click **Get External Data** on the **Data** tab of the ribbon, click **From Other Sources**, and then select **From Analysis Services**. In the Data Connection Wizard, type the Analysis Services server name and provide credentials if you are not using Windows Authentication to connect with your own account. On the next page of the wizard, select the database in the drop-down list and then select the cube to query in the list of cubes and perspectives, as shown in Figure 93. On the final page of the wizard, change the file name and provide a friendly name if desired, and save the Office Data Connection (ODC) file to your local machine or use the **Browse** button to save it to a network or a SharePoint data connections library.



Select Database and Table

Select the Database and Table/Cube which contains the data you want.

Select the database that contains the data you want:

SSAS Succinctly

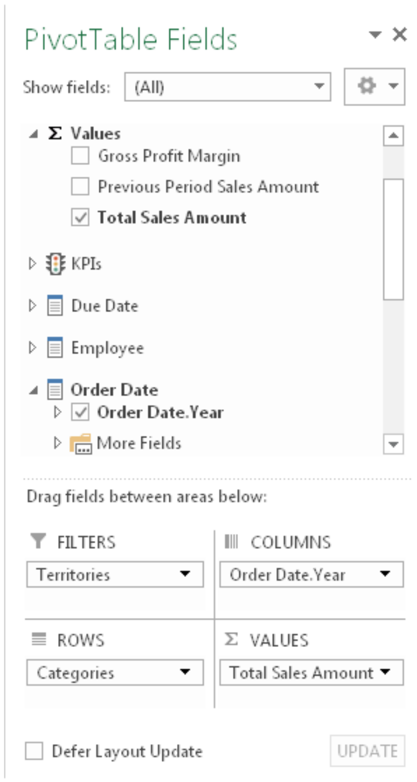
[✓] Connect to a specific cube or table:

| Name | Description | Modified | Created | Type |
|---|---|---|---|---|
| Reseller Sales by Order Date | | 4/18/2013 10:02:12 AM | | PERSP |
| Sales | | 4/18/2013 10:02:12 AM | | CUBE |

*Figure 93: Analysis Services Data Connection in Excel*

***Note: Although these instructions and screenshots are specific to Excel 2013, similar functionality is available in Excel 2007 and Excel 2010.***

In the **Import Data** dialog box that appears, you can choose to create a PivotTable Report, a PivotChart, or a Power View Report. For this example, let's say you want to create a PivotTable. On the right side of the screen, the **PivotTable Fields** list is displayed with measures organized by measure group at the top of the list. You can scroll through the list to locate dimensions, with user-defined hierarchies listed first and attribute hierarchies listed under the More Fields folder. You add fields to the report by selecting the field's check box. As you make your selections, measures are displayed in the Values area, non-date attributes are displayed in the Rows area, and date attributes are displayed in the Columns area, as shown in Figure 94. If you want to add a filter, you must drag the field from the field list to the Filters area.


PivotTable Fields

Show fields: (All)

Σ Values
- ☐ Gross Profit Margin
- ☐ Previous Period Sales Amount
- ☑ Total Sales Amount

KPIs

Due Date

Employee

Order Date
- ☑ Order Date.Year
- More Fields

Drag fields between areas below:

| FILTERS | COLUMNS |
|---|---|
| Territories | Order Date.Year |

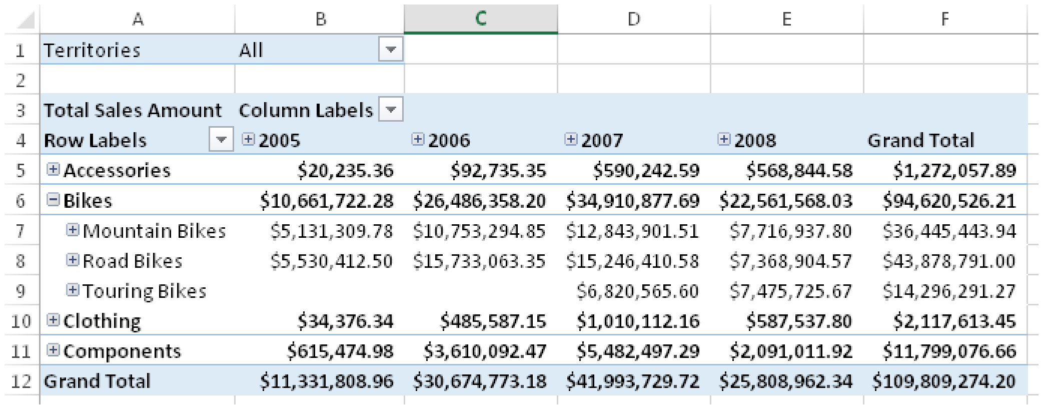
| ROWS | VALUES |
|---|---|
| Categories | Total Sales Amount |

☐ Defer Layout Update

UPDATE

*Figure 94: PivotTable Fields List in Excel*

As you make selections (unless you select the **Defer Layout Update** check box at the bottom of the fields list), a query executes to retrieve data from the cube and display the results according to the layout you define in the fields list. If you add a user-defined hierarchy to rows or columns, you can click the plus sign next to a label to view the data in the level below, as shown in Figure 95. If you add a filter to the report, you can use the filter's drop-down list above the pivot table to make a new selection and update the contents of the report.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Territories | All | | | | |
| 2 | | | | | | |
| 3 | Total Sales Amount | Column Labels | | | | |
| 4 | Row Labels | ⊞2005 | ⊞2006 | ⊞2007 | ⊞2008 | Grand Total |
| 5 | ⊞Accessories | $20,235.36 | $92,735.35 | $590,242.59 | $568,844.58 | $1,272,057.89 |
| 6 | ⊟Bikes | $10,661,722.28 | $26,486,358.20 | $34,910,877.69 | $22,561,568.03 | $94,620,526.21 |
| 7 | ⊞Mountain Bikes | $5,131,309.78 | $10,753,294.85 | $12,843,901.51 | $7,716,937.80 | $36,445,443.94 |
| 8 | ⊞Road Bikes | $5,530,412.50 | $15,733,063.35 | $15,246,410.58 | $7,368,904.57 | $43,878,791.00 |
| 9 | ⊞Touring Bikes | | | $6,820,565.60 | $7,475,725.67 | $14,296,291.27 |
| 10 | ⊞Clothing | $34,376.34 | $485,587.15 | $1,010,112.16 | $587,537.80 | $2,117,613.45 |
| 11 | ⊞Components | $615,474.98 | $3,610,092.47 | $5,482,497.29 | $2,091,011.92 | $11,799,076.66 |
| 12 | Grand Total | $11,331,808.96 | $30,674,773.18 | $41,993,729.72 | $25,808,962.34 | $109,809,274.20 |

*Figure 95: PivotTable Fields List in Excel*

***Tip: You can add multiple fields to the Filters, Columns, Rows, or Values area of the layout section in the PivotTable Fields List. After adding a field to the layout section, you can also drag it to a different area to rearrange the report layout.***

## Microsoft SQL Server Reporting Services

Reporting Services is a business intelligence component included as part of SQL Server to support the reporting life cycle, from report development to management to user access. Although reports can be interactive, either by allowing users to filter content through changing report parameter values or to drill from summary to detail data in the same or separate reports, reports are much more static and less interactive than with an exploratory tool like Excel. You must plan in advance the type of interactivity you want to support, and then explicitly build that interactivity.

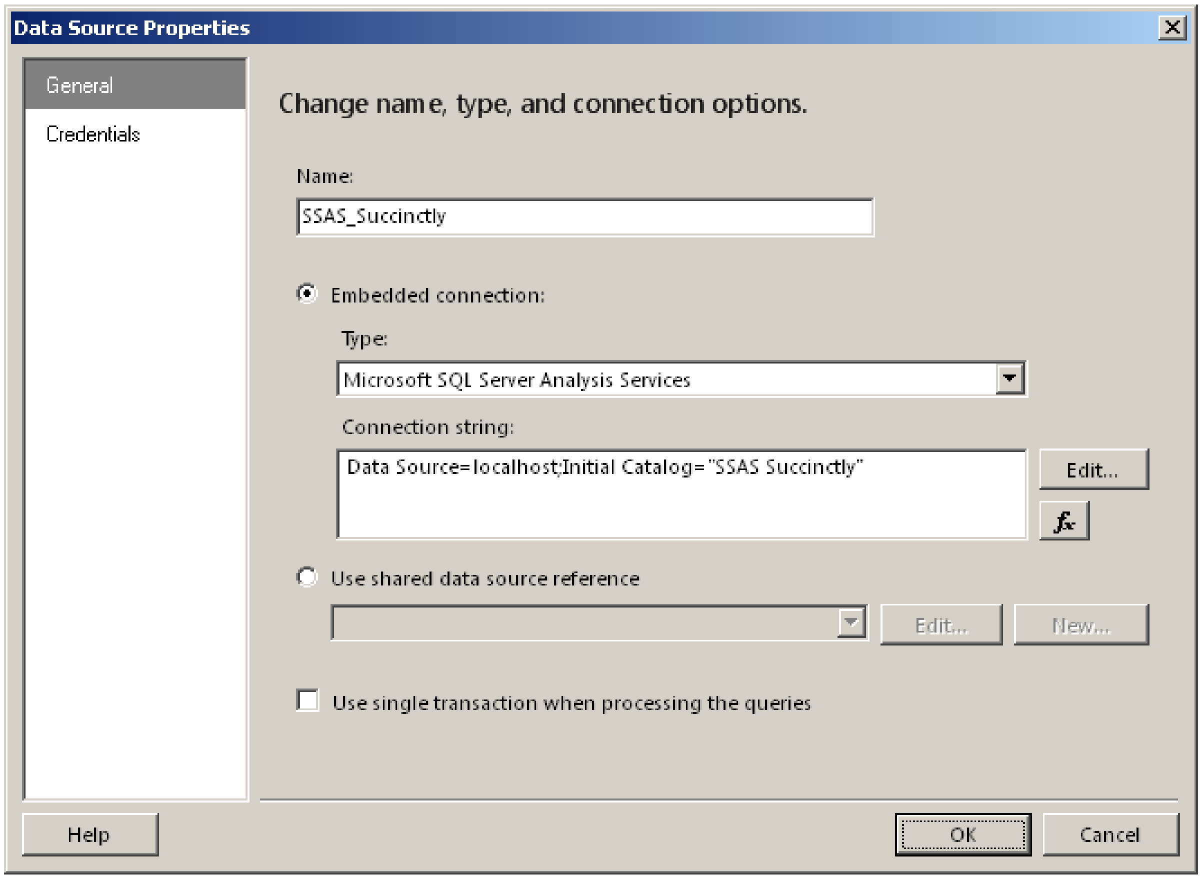
On the other hand, Power View in SharePoint is a new feature in the latest version of Reporting Services in SharePoint integrated mode that supports interactive, ad hoc reporting. The initial release did not support connections to Analysis Services in multidimensional mode, but SQL Server 2012 Service Pack 1 Cumulative Update 4 adds this capability. However, Power View in Excel 2013 is not currently compatible with multidimensional Analysis Services.

To create reports that can be published to a Reporting Services server, you can use either of two tools: Report Designer in SSDT or Report Builder. Report Designer is a tool intended for the professional report developer who often works with multiple reports at once, whereas Report Builder is intended for the less technically savvy user who works with one report at a time. You can access Report Designer by creating a report server project in SSDT, but you download Report Builder from the Report Manager web application (typically at http://<server>/reports). Working with the layout in each environment is similar, but the user interface differs slightly between the two. In Report Designer, you use menus, toolbars, and a toolbox to perform commands and work with report items, but in Report Builder you use a ribbon interface much like you do in Office products. Report Builder also includes several wizards to walk you through the process of building items in a report.

***Note: This section includes only basic information for connecting to Analysis Services as a data source. Report development in Reporting Services is covered in more detail in the Windows Azure SQL Reporting Succinctly e-book available from Syncfusion. Although the focus of the e-book is the Windows Azure version of reporting, the functionality described for report development also applies to the on-premises version of SQL Server Reporting Services.***

### Report Designer

To create a new report in Report Designer, right-click the **Reports** folder of a report server project in the **Solution Explorer** window, point to **Add**, and then click **New Item**. In the **Report Data** folder, right-click the **Data Sources** folder and select **Add Data Source**. In the **Data Source Properties** dialog box, provide a name for the data source, select **Microsoft SQL Server Analysis Services** in the **Type** drop-down list, and provide a connection string (or use the **Edit** button to open the user interface to generate the connection string), as shown in Figure 96.



Change name, type, and connection options.

Name: SSAS_Succinctly

(●) Embedded connection:

Type: Microsoft SQL Server Analysis Services

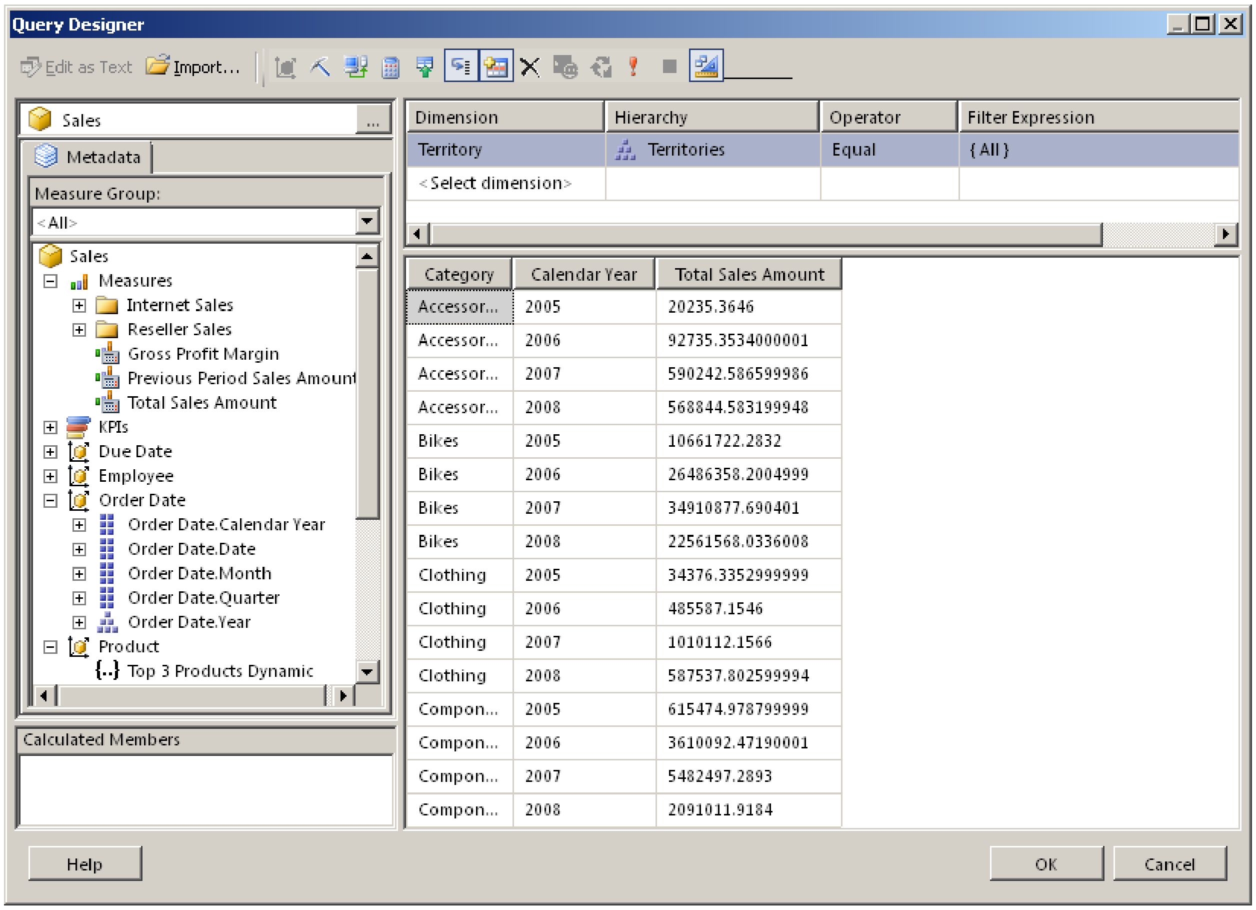
Connection string: Data Source=localhost;Initial Catalog="SSAS Succinctly"

( ) Use shared data source reference

☐ Use single transaction when processing the queries

*Figure 96: Analysis Services Data Connection in Report Designer*

Next, create a dataset to hold the query definition. Right-click the **Datasets** folder in the **Report Data** pane, and select **Add Dataset**. In the **Dataset Properties** dialog box, select the **Use A Dataset Embedded In My Report** radio button, select the data source in the drop-down list, and click **Query Designer**. In the Query Designer, you can select a different perspective or cube by clicking the ellipsis button in the top left corner and changing the current selection. Then you can drag items from the metadata panel on the left into the query data panel on the right (if you have a simple query), as shown in Figure 97. You can add a dimension to the filter pane at the top of the query designer, and select the **Parameters** check box at the far right of the row to allow the user to interactively change the filter value at report run-time.



| Dimension | Hierarchy | Operator | Filter Expression |
|---|---|---|---|
| Territory | Territories | Equal | { All } |
| <Select dimension> | | | |

| Category | Calendar Year | Total Sales Amount |
|---|---|---|
| Accessor... | 2005 | 20235.3646 |
| Accessor... | 2006 | 92735.3534000001 |
| Accessor... | 2007 | 590242.586599986 |
| Accessor... | 2008 | 568844.583199948 |
| Bikes | 2005 | 10661722.2832 |
| Bikes | 2006 | 26486358.2004999 |
| Bikes | 2007 | 34910877.690401 |
| Bikes | 2008 | 22561568.0336008 |
| Clothing | 2005 | 34376.3352999999 |
| Clothing | 2006 | 485587.1546 |
| Clothing | 2007 | 1010112.1566 |
| Clothing | 2008 | 587537.802599994 |
| Compon... | 2005 | 615474.978799999 |
| Compon... | 2006 | 3610092.47190001 |
| Compon... | 2007 | 5482497.2893 |
| Compon... | 2008 | 2091011.9184 |

*Figure 97: Analysis Services Query Designer*

***Tip: You can click the last button in the Query Designer toolbar to switch the design mode from graphical to text. In text design mode, you can write the MDX query to generate the desired results. However, unlike an MDX query that you can write in SSMS or other third-party tools, your query must reference only measures on the COLUMNS axis. You can use non-measure dimensions only on the ROWS axis of your query.***

After creating the dataset, you can design the report layout. For example, you can create a chart from the cube data by dragging the Chart report item from the toolbox to the report design surface. Select a chart type, such as a column chart, and then click the chart placeholder that appears on the design surface to open the **Chart Data** pane. Click the plus sign that appears in each section of the **Chart Data** pane to assign a field to that section, as shown in Figure 98. Click the **Preview** tab to execute the dataset query and view the rendered report. Further adjustments can be made to fine-tune the appearance of the chart, such as changing the chart and axis titles, changing the color palette of the chart, or resizing the chart, to name just a few options.

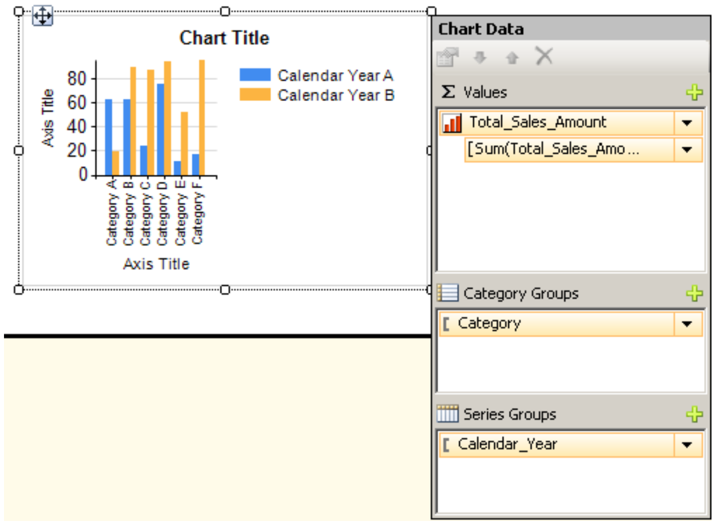


*Figure 98:  Chart Report Item in Report Designer*

### Report Builder

After launching Report Builder, you can bypass the wizards prompting you to build a new report and develop a new report much like you can in Report Designer. You follow the same steps described in the previous section to add a new data source and a new dataset. Once the dataset is created, you can create any type of report item by using the **Insert** tab of the ribbon—a table, matrix, list, or chart, for example—and add fields from the dataset to that report item. To preview the report, click **Run** on the **Home** tab.

### Power View in SharePoint

If you have a report server configured in SharePoint integrated mode, and have installed the required service pack, you can use Power View to interactively explore data in a multidimensional Analysis Services database. You can learn more about configuring the environment and working with Power View by reviewing the following TechNet article: SQL Server 2012 with Power View for Multidimensional Models.

***Note: At the time of this writing, Power View for Multidimensional Models is available only as part of a cumulative update and distributed as a hotfix. Accordingly, it may not be suitable for a production environment until it is released in the next service pack for SQL Server 2012.***

## Microsoft SharePoint Server

SharePoint Server 2013 (like its predecessors SharePoint Server 2010 and Microsoft Office SharePoint Server 2007) includes several business intelligence features that support connections to Analysis Services:

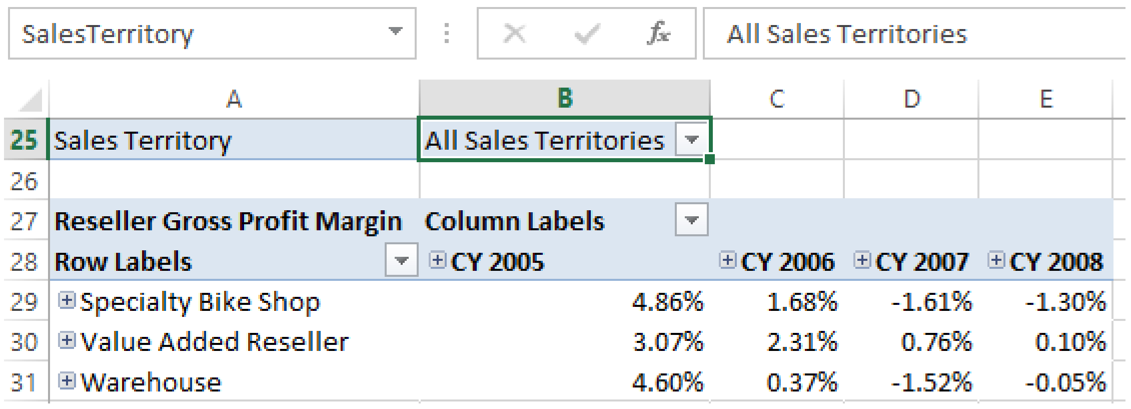
- Excel Services
- Dashboard Filters
- PerformancePoint Services

## Excel Services

You can publish an Excel workbook containing pivot tables and pivot charts to SharePoint to enable multiple users to interact with the same workbook. In SharePoint Server 2013, much of the same interactivity available in the desktop client application is also available in the browser. In earlier versions of SharePoint, users can only drill, filter, or sort, but cannot change the contents of the pivot table or pivot chart.

When creating an Excel workbook that you intend to publish, it is a good idea to publish the ODC file associated with the Analysis Services connection to SharePoint as a separate file rather than embed the connection information in the workbook. To modify the connection, click **Connections** on the **Data** tab of the ribbon, select the applicable connection, and then click **Properties**. In the **Connection Properties** dialog box, click the **Definition** tab, and then click **Export Connection File**. In the **File Save** dialog box, type the URL for the Data Connections library in the **File Name** box, press **Enter**, and then type a name for the ODC file to save it. A **Web File Properties** dialog box is displayed asking for a title, description, and keywords for the ODC file. After you save the file, you can select the **Always Use Connection File** check box in the **Connection Properties** dialog box in Excel.

If a pivot table has a filter defined, you can convert the filter to a parameter so that you can connect it to a filter in a dashboard if necessary. To do this, click the workbook cell containing the filter value and then in the box to the left of the formula bar, replace the cell reference by typing a name for the cell, as shown in Figure 99.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 25 | Sales Territory | All Sales Territories | | | |
| 26 | | | | | |
| 27 | Reseller Gross Profit Margin | Column Labels | | | |
| 28 | Row Labels | CY 2005 | CY 2006 | CY 2007 | CY 2008 |
| 29 | Specialty Bike Shop | 4.86% | 1.68% | -1.61% | -1.30% |
| 30 | Value Added Reseller | 3.07% | 2.31% | 0.76% | 0.10% |
| 31 | Warehouse | 4.60% | 0.37% | -1.52% | -0.05% |

*Figure 99: Cell Name Assignment*

When you are ready to publish the workbook to Excel Services, click **File** > **Save As** > **SharePoint**, and then click **Browse**. Navigate to the document library in which you want to store the workbook, type a name for the workbook, but before you click **Save**, click **Browser View Options** in the **Save As** dialog box. In the **Browser View Options**, click the **Parameters** tab, click **Add**, and then select the check box next to the cell name you defined for the filter. This cell name becomes the parameter that you reference when you create a dashboard filter as described in the next section of this chapter.

## Dashboard Filters

If you create a dashboard (also known as a Web Part Page) in SharePoint, you can combine content from various sources that in turn draw data from Analysis Services. For example, you might create a dashboard that includes any combination of Reporting Services reports, Excel workbooks, and PerformancePoint content such as scorecards, analytic charts, or analytic grids. If these items use Analysis Services multidimensional databases as a source, you can add an SQL Server Analysis Services Filter Web Part to the dashboard and thereby allow users to filter the related content with one selection.

Let's assume that you have an existing dashboard in SharePoint 2013 with a Reporting Services report and an Excel workbook in it, both of which have the same Analysis Services database as a source, and you have permission to edit the dashboard. Also assume that both the report and the workbook have a parameter configured to filter by Sales Territory. Open the dashboard, click the **Page** button in the ribbon, and then click **Edit Page**. In one of the dashboard zones, click the **Add a Web Part** link, click **Filters** in the **Categories** list, select **SQL Server Analysis Services Filter** in the **Parts** list, and then click **Add**. To configure the filter, click the **Open the Tool Pane** link in the newly added Web Part. Assuming you do not have a Status List Web Part on the page, select **SharePoint Data Connection Library** as the data connection source, click the button to the right of the data connection text box, navigate to the **Data Connections** folder, select the ODC file associated with the Analysis Services database that the report and workbook have in common, and select **Insert**. If you don't have an Analysis Services database yet, follow the instructions in the Microsoft Excel section of this chapter to create one and save it to the **Data Connections** folder in SharePoint.

In the **Dimension** drop-down list, select the **Sales Territory** dimension. Next, in the **Hierarchy** drop-down list, select the user-defined hierarchy or the attribute hierarchy that matches the hierarchy used in the report and workbook parameters, and then click **OK** to save the configuration and close the tool pane.

Your next step is to connect the filter to the report and workbook. To do this, click the arrow icon that appears in the top right corner of the **Filter Web Part**. (You might need to click the Web Part to display the icon.) In the submenu, point to **Connections** > **Send Filter Values To**, and then click the report title, as shown in Figure 100.

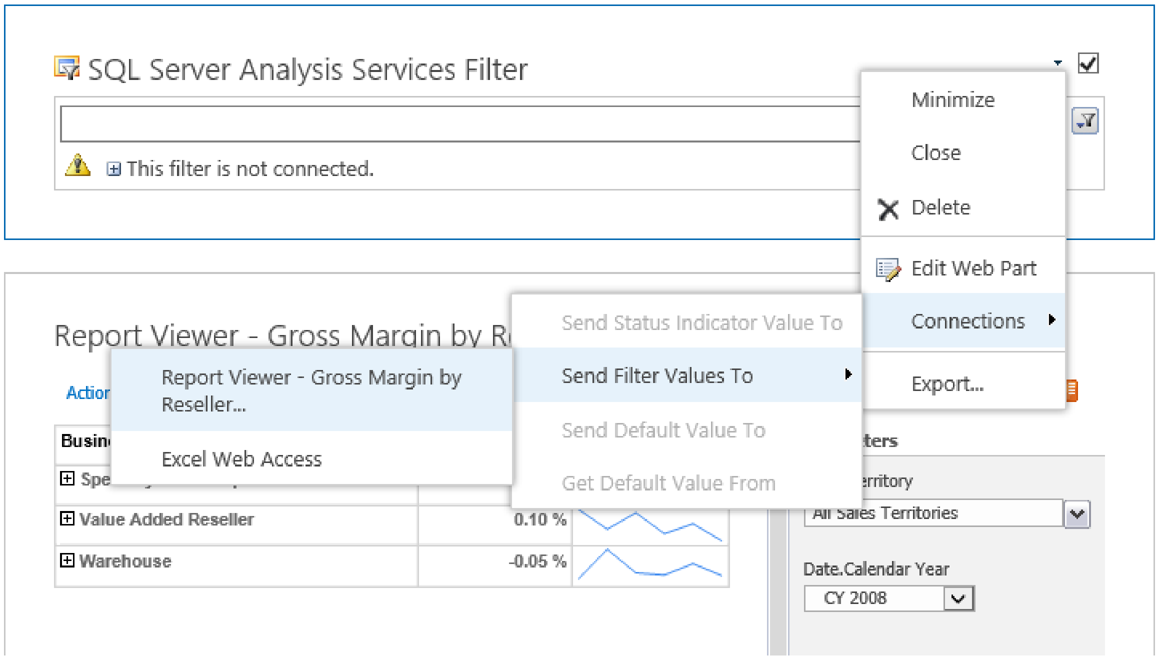


*Figure 100: Connection between Filter Web Part and Report Viewer Web Part*

In the **Configure Connection** dialog box, select the parameter to which the filter value applies and click **Finish**. Repeat the previous steps to connect the filter to the workbook. In the case of the workbook, the dialog box that is displayed asks for a connection, which you should set as **Get Values for Multiple Parameters**. Click the **Configure Connection** tab in the dialog box, select the applicable parameter in the workbook, and click **Finish**. Click the **Stop Editing** button in the ribbon to finalize your changes, and then click the **Browse** button to review the finished dashboard. You can test the filter by clicking the icon to the right of the filter box and making a selection of one or more dimension members. When you click **OK**, the filter is applied to the report and the workbook.

## PerformancePoint Services

PerformancePoint Services is a shared service application available in SharePoint that allows you to create scorecards and analytical reports that use Analysis Services data as a source. You can create a multipage dashboard in PerformancePoint Services that you can store in a SharePoint dashboard library or you can use Web Parts to display the scorecards and analytical reports in a standard SharePoint dashboard. You can learn more about working with PerformancePoint Services by reviewing Create Dashboards by using PerformancePoint Services (SharePoint Server 2013).

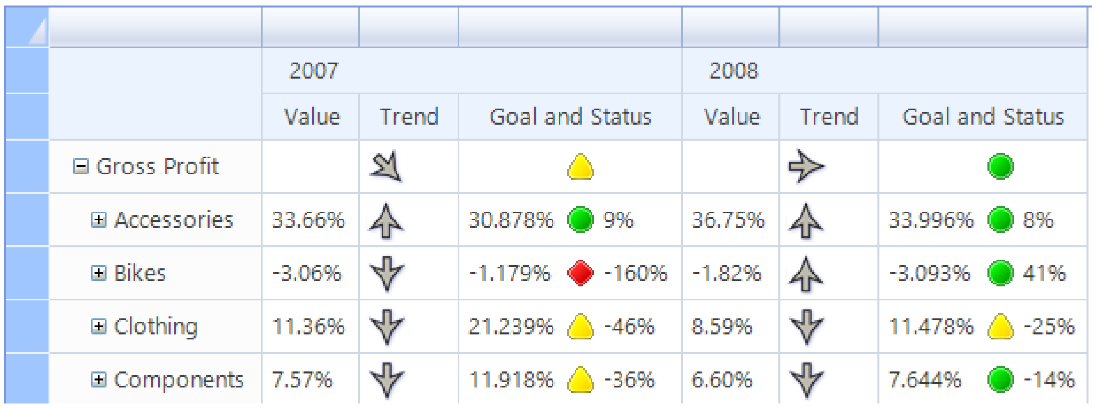
Regardless of the type of dashboard you want to create, you start by creating a data source object in PerformancePoint Services. To do this, click the **PerformancePoint Content** link in the **Quick Launch** panel on the left side of the SharePoint site, open the **PerformancePoint** tab on the ribbon, and click the **Dashboard Designer** button. On the **Create** tab of the **Dashboard Designer** ribbon, click **Data Source**, which is available only when you have the Data Connections folder selected in the Workspace Browser pane. Select the **Analysis Services** template and click **OK**. In the **Connection Settings**, type the server name, select the database in the drop-down list, and then select the cube in the next drop-down list.

Next, select an authentication method. Make sure the cube security grants read permission to the unattended service account for PerformancePoint Services if you keep the default setting here. Otherwise, you can set up an application ID to use Secure Store Service for the stored account option, or configure user security to use the per-user identity option. You can provide a name for the data source by editing the name in the **Workspace Browser** or by typing it in the **Properties** tab.

On the **Time** tab, you map the members of your Date dimension (called a Time dimension in most Microsoft products) to the standard references that PerformancePoint Services understands. Select the applicable dimension in the **Time Dimension** drop-down list. Then, click the **Browse** button in the **Reference Member** section and choose a date for the beginning of the year that exists in the Date dimension. Any year suffices. In the **Hierarchy Level** drop-down list, select the level to which the date you selected belongs. Next, in the **Reference Date** section, click the icon to use a calendar to find a date that corresponds to the dimension member you selected. Lastly, map each member level from your date dimension to a standard time aggregation description, such as Year, Quarter, Month, and so on. When finished, click the **Save** icon above the ribbon to store the data source in the PerformancePoint Services content library in SharePoint.

Now you're ready to use the data source. Click **PerformancePoint Content** in the Workspace Browser. To create a scorecard using KPIs defined in the selected cube, click **Scorecard** on the **Create** tab of the ribbon and then select the **Analysis Services** template. In the wizard, select the data source you created, click **Next**, and then select the option to import the KPIs. Select the check box for each KPI to import and click **Next**. You can optionally add filters to modify the measures in the KPI and define members to display on the scorecard's columns. A basic scorecard will be displayed when you complete the wizard.

You can expand the **Dimensions** node in the **Details** pane to locate dimensions to add to the scorecard. To achieve the result shown in Figure 101, drag the **Categories** user-defined hierarchy to the **Gross Profit** cell in the scorecard, but position the cursor toward the right edge of the cell to display the "Last child" tooltip, and then release the mouse button. In the **Select Members** dialog box, select the individual categories to include in the scorecard and click **OK**. On the **Edit** tab of the ribbon, click the **Update** button to display the KPI values by category. You can expand each category to drill down to the subcategory and then to the product level to view additional KPI values.



| | 2007 | | | 2008 | | |
|---|---|---|---|---|---|---|
| | Value | Trend | Goal and Status | Value | Trend | Goal and Status |
| ⊟ Gross Profit | | ↘ | ▲ | | → | ● |
| ⊞ Accessories | 33.66% | ↑ | 30.878% ● 9% | 36.75% | ↑ | 33.996% ● 8% |
| ⊞ Bikes | -3.06% | ↓ | -1.179% ◆ -160% | -1.82% | ↑ | -3.093% ● 41% |
| ⊞ Clothing | 11.36% | ↓ | 21.239% ▲ -46% | 8.59% | ↓ | 11.478% ▲ -25% |
| ⊞ Components | 7.57% | ↓ | 11.918% ▲ -36% | 6.60% | ↓ | 7.644% ● -14% |

*Figure 101: Scorecard with Analysis Services KPIs*

***Note: If you get an error during the KPI import process, you might need to install the Microsoft Analysis Management Objects (AMO) from SQL Server 2008. If the version 10.0.0.0 of Microsoft.AnalysisServices does not exist in the Windows\assembly folder on the SharePoint server, you can download the AMO installer on the Microsoft SQL Server 2008 Service Pack 2 Feature Pack page after expanding the Install Instructions section. After installation completes successfully, open a command prompt as an administrator and run iisreset.***

***Tip: Notice that an item appears in the PerformancePoint Content folder for each KPI that you import. You can configure the format for the value, goal, and trend amounts by selecting the KPI and then clicking the respective Number Format cell in the editor.***

***You can also control whether a column displays an image, a value for trend or status, or a variance. Right-click the cell in the scorecard and select Metric settings on the Values page of the dialog box. You can specify the name of the metric, and the KPI amount—such as actual, target, trend, or variance—to display.***

Another way to interact with an Analysis Services cube in PerformancePoint is to create an analytic chart or analytic grid. In fact, although you can use many different data source types for a scorecard, you can use only an Analysis Services data source for the analytic reports. The process to create either of these reports is similar. On the **Create** tab of the Dashboard Designer, select the button for the type of report you want to create. In the example that follows, I use Analytic Chart. Next, select the data source for Analysis Services. You then drag dimensions and measures to the Series, Bottom Axis, or Background areas below the report layout, as shown in Figure 102. For the Bottom Axis, the default report displays the **All** level of the hierarchy. To view the next level in the hierarchy, you must click the arrow icon next to the hierarchy name to display the **Select Members** dialog box, and then select the members to display. An alternative to explicitly selecting members is to right-click the **All** member and then click **Select Children**. That way if new members are added later to the dimension, those members automatically appear in the report. Be sure to clear the selection of the Default Member.

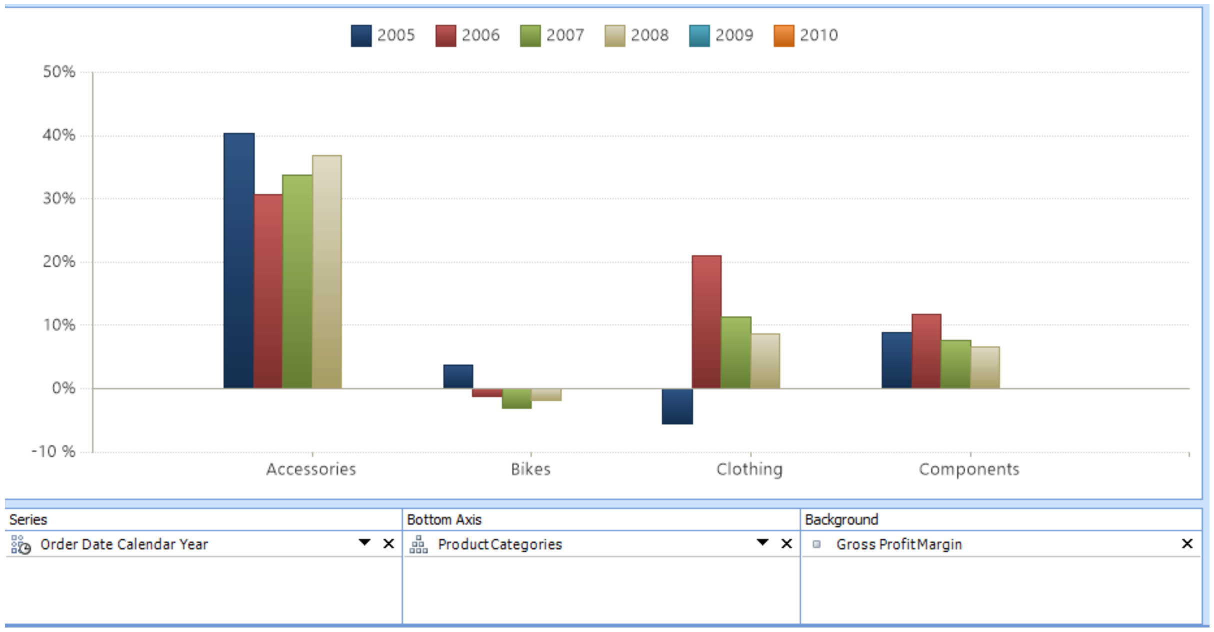


*Figure 102: Analytic Chart*

***Tip: You can right-click the report and select the Format Report option to reposition the legend. It can appear across the top of the report or along the right edge of the report.***

An analytic report is interactive, allowing you to drill down or pivot the report. You can double-click a label in the legend or on the horizontal axis if the label is part of a user-defined hierarchy. You can also right-click the chart item (such as a bar) and choose the option to **Drill Down To** when you want to switch the current view to an alternate dimension, or you can choose the option to **Show Only** or **Remove** when you want to filter the report to keep or hide selected items. Sorting and filtering options are also available on the submenu when you right-click the report. You can even change a bar chart to a grid, line chart, or pie chart by right-clicking the report, selecting **Report Type**, and choosing one of the available options.

## Custom Applications (ADOMD.NET)

If you want to embed reporting and analysis using Analysis Services as a data source in a custom application, you can use the ADOMD.NET object model. If you have experience with ADO.NET, working with ADOMD.NET will be very similar. You can build an interface that allows the user to interactively explore a cube, or you can embed an MDX query into your application. In this example, I show you how to build a simple Windows Form application in Visual Basic to query a cube.

The first step is to create a new **Windows Forms** project in Visual Studio. Then, drag a **DataGridView** control from the **Toolbox** onto the form. In the **DataGridView** task list, click the **Dock in Parent Container** link to complete the form.

Next, you need to add a reference to the ADOMD.NET client library. To do this, right-click your project name in **Solution Explorer**, select **Add Reference**, click the **Browse** tab, navigate to the **Program Files\Microsoft.NET\ADOMD.NET\110** folder, and double-click **Microsoft.AnalysisServices.AdomdClient.dll**.

Now it's time to add code that uses the ADOMD.NET library. Double-click the title bar of the form to create an event handler for the load event. Add the following code as the first line (above the class declaration):

```
Imports Microsoft.AnalysisServices.AdomdClient
```

In the load event, add the following code to create a simple connection string that uses Windows authentication and assigns it to an AdomdConnection object:

```
Dim ssasConn As New AdomdConnection("Data Source=localhost;Catalog=SSAS
Succinctly")
```

Next, add in the code to define an MDX query that you assign to an AdomdDataAdapter. An ADOMD.NET data adapter is similar to the ADO.NET data adapter except that it requires an AdomdConnection object as the second argument on creation.

```
Dim dataAdapter As New AdomdDataAdapter("select [Measures].[Gross Profit
Margin] on columns, [Product].[Category].[Category].Members on rows from
[Sales]", ssasConn)
```

Now you need a dataset that you fill from the **AdomdDataAdapter**:

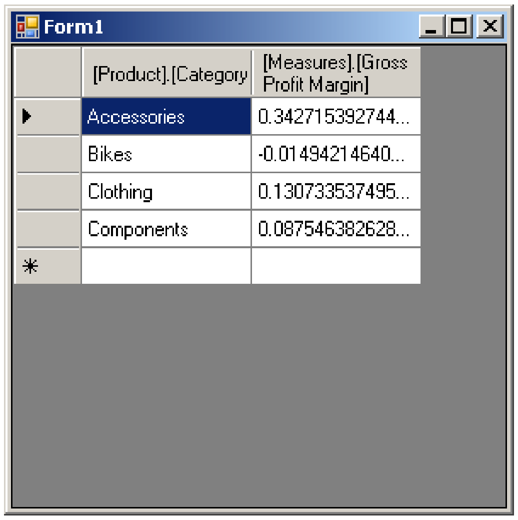
```
Dim ds As New DataSet()
dataAdapter.Fill(ds)
```

And lastly, you assign the results to the **DataGridView** control to display in your form:

```
DataGridView1.DataSource = ds.Tables(0)
```

***Note: Although ADO.NET and ADOMD.NET are similar frameworks and use similar objects, the [CellSet]() object is unique to ADOMD.NET. You can store not only the results of an MDX query in this object, but also metadata for the objects returned by the query. You can then create an interface that allows the user to explore a cube in a manner similar to a pivot table or the query designer in Reporting Services.***

Press **F5** to run the application, and view the results as shown in Figure 103. Of course, you can do many more complex operations once you have the data stored in a data adapter. You can also format the data more nicely. Nonetheless, this simple example provides the key elements of code necessary to access data from an Analysis Services database.



| | [Product].[Category] | [Measures].[Gross Profit Margin] |
|---|---|---|
| ▶ | Accessories | 0.342715392744... |
| | Bikes | -0.01494214640... |
| | Clothing | 0.130733537495... |
| | Components | 0.087546382628... |
| * | | |

*Figure 103: Sample ADOMD.NET Application*

***Note: To deploy your application, you must include the ADOMD.NET redistributable package as part of your application's installation process. You can download the package from the [Microsoft SQL Server 2012 Feature Pack]() page after expanding the Install Instructions section.***