Analysis Services

Analysis Services Overview

Microsoft® SQL Server™ 2000 Analysis Services includes online analytical processing (OLAP) and data mining. Use this table to acquaint yourself with this release of Analysis Services documentation.

Section	Description
What's New in Analysis Services	Describes the enhancements and new features in Analysis Services with links to information in the documentation.
Installing Analysis Services	Provides instructions for installing Analysis Services.
Analysis Services Architecture	Describes the features of Analysis Services, the server and client architecture, the object architecture, and security.
Data Warehousing and OLAP	Describes the use of OLAP in data warehouses.
Administering Analysis Services	Describes the tools and tasks used in administering Analysis Services.
Administrator's Reference	Provides reference information for Analysis Services, including specifications and limits, the use of SQL in Analysis Services, the help topics for the user interface, and the compliance of Analysis Services with the OLAP and data mining specifications of OLE DB.
MDX	Provides introductory material, reference information, and examples that detail the use of Multidimensional Expressions (MDX) in Analysis Services.
Analysis Services Troubleshooting	Provides information to assist in resolving problems that may occur in Analysis Services.
Programming Analysis Services Applications	Describes the programming information and reference material provided for

	developing Analysis Services administrative and client applications.
How To	Provides the administrative procedures for Analysis Services.

Analysis Services

What's New in Analysis Services

Microsoft® SQL Server™ 2000 extends and renames the former OLAP Services component, now called Analysis Services. Many new and improved features significantly enhance the analysis capabilities of the acclaimed OLAP Services introduced in SQL Server version 7.0. In this release, Analysis Services introduces data mining, which can be used to discover information in OLAP cubes and relational databases.

The What's New topics contain brief overviews of the new Analysis Services features with links to the conceptual topics that discuss each feature in more detail and provide further links into the documentation.

Topic	Description
Cube Enhancements	New cube types and enhanced cube functionality substantially extend the scalability and functionality of Analysis Services.
Dimension Enhancements	New dimension and hierarchy types, features, and improvements extend the analysis capabilities of cubes.
Data Mining Enhancements	New in this release, data mining is integrated into online analysis and can be used to discover information in OLAP cubes and relational databases.
Security Enhancements	Security enhancements include using roles on cube cells and dimension members, additional authentication methods, and improved enforcement.
Client Connectivity Enhancements in PivotTable Service	Client applications can use many new features and enhancements such as data mining, HTTP or HTTPS connections, additional dimension types, and cell allocation for writeback.
Other Enhancements	Other enhancements provide a variety of new features including multiuser administration, MDX Builder, additional Multidimensional

	Expressions (MDX) functions, Virtual Cube Editor, support for Active Directory™, and more.

Analysis Services

Cube Enhancements

Microsoft® SQL Server™ 2000 Analysis Services substantially extends the scalability and functionality of OLAP cubes. You can distribute cube data across multiple servers to provide more storage capacity, create linked cubes to distribute end-user access to information without duplicating cube data, create cubes that are updated in real time as data changes, and use a number of other new features to create cubes that address your specific business needs.

Distributed Partitioned Cubes

You can create distributed partitioned cubes by using remote partitions that distribute a cube's data among multiple Analysis servers. A distributed partitioned cube is administered on a central Analysis server. For more information, see Remote Partitions.

Real-Time OLAP

Real-time OLAP provides a multidimensional OLAP view of data that is continually updated as the underlying data changes. Real-time cubes implement real-time OLAP by using ROLAP storage for partitions and dimensions, new SQL Server 2000 indexed views for aggregations, and automatic notification by the SQL Server 2000 relational engine when data changes. Real-time cubes provide the capability to develop new categories of OLAP solutions such as call-center management, stock market analysis, or campaign management. For more information, see Real-Time Cubes.

Linked Cubes

A cube can be stored on a single Analysis server and then defined as a linked cube on other Analysis servers. End users connected to any of these Analysis servers can then access the cube. This arrangement avoids the more costly alternative of storing and maintaining copies of a cube on multiple Analysis servers. Linked cubes can be connected using TCP/IP or HTTP. To end users, a linked cube looks like a regular cube. For more information, see Linked Cubes.

Indexed Views for Aggregations

Indexed views for increased performance and flexibility are used instead of aggregation tables for ROLAP partitions if the partition's source data is stored in SQL Server 2000 and if certain criteria are met. For more information, see Indexed Views for ROLAP Partitions.

Cube Processing

You can use lazy aggregations to make cube data available to end users while aggregations are being calculated. When processing cubes for which the underlying data contains dimension key errors, you can elect to stop processing on key errors, stop processing after a specified number of errors, or ignore all key errors. You can have errors logged to a file for later review. For more information, see Processing Cubes.

Calculated Cells

You can specify formulas that apply to individual cells or to sets of cells in a cube. These formulas can contain conditional calculations that compute a new value for a cell or set of cells based on values in the cell or cells, or on values in other cells in the cube. Calculated cells use Multidimensional Expressions (MDX) expressions and you can specify calculations to be performed in multiple passes. Calculated cells can be used in complex financial modeling and budgeting applications; for example, you can specify a default value such as a percentage of a parent cell if the cell value is zero, or to use the actual value if it is not zero. For more information, see Calculated Cells.

Drillthrough

Client applications that support drillthrough can now allow end users to select a cube cell and retrieve a result set from the source data for that cell. You can use roles to control user access to the drillthrough functionality. For more information, see Specifying Drillthrough Options.

Actions

Actions enable end users to act upon the outcomes of their analyses. An action is a predefined operation that an end user can initiate upon a selected cube or portion of a cube. The operation can start an application with the selected item as a parameter or retrieve information about the selected item. A wizard is provided to help you create actions. For more information, see Actions.

Distinct Count

You can use the new **DistinctCount** aggregate function to analyze the number of unique occurrences of events or transactions in your data, such as unique users visiting a Web site. For more information, see **DistinctCount**.

Hidden Cube Elements

You can hide complete cubes, dimensions, levels, measures, or member properties from end users who browse cubes with client applications. The visibility of these objects is controlled by the **Visible** property. For more information, see Properties Pane (Cube Editor Data View) and Properties Pane (Dimension Editor Data View).

Named Sets

You can create, name, and save sets of dimension members or set expressions in a cube. Client applications can use a named set like a dimension by placing the named set on an axis. For more information, see Named Sets.

Default Measures

Default measures can be specified for each cube and varied by role to control end users' default views of cubes. For more information, see Properties Pane (Cube Editor Data View) and Custom Rules in Dimension Security.

Virtual Cube Editor

This new editor for virtual cubes is similar in function to Cube Editor. Virtual Cube Editor replaces and expands the functionality provided by the Calculated Member Manager Add-in for SQL Server version 7.0 OLAP Services. For more information, see Virtual Cube Editor.

Analysis Services

Dimension Enhancements

This release of Microsoft® SQL Server™ 2000 Analysis Services adds significant functionality to OLAP analysis with a number of new dimension types, features, and improvements.

Parent-Child Dimensions

A new parent-child dimension type supports hierarchies based on parent-child links between members in columns in a source table. Such hierarchies represent structures that include organization charts and part assemblies. Data members can be used to provide data for nonleaf members, such as the direct commission amount for a sales department manager or the individual salaries for all members in an organizational chart. For more information, see Parent-Child Dimensions.

ROLAP Dimensions

Extremely large dimensions can now be accommodated using the ROLAP storage mode. The dimension data remains in the database table and is not subject to the size limitations of MOLAP. For more information, see Dimension Storage Modes.

Write-Enabled Dimensions

The members of write-enabled dimensions can be updated through Analysis Manager and client applications that support dimension writeback. Roles are used to control dimension write access by client applications. For more information, see Write-Enabled Dimensions.

Changing Dimensions

The new changing dimension type permits dimension members to be deleted, moved, added, or renamed without requiring the cube to be fully processed after changes. This increases the availability of cubes to client applications. For more information, see Changing Dimensions.

Dependent Dimensions

Dependent dimensions permit improved aggregation design optimization by using knowledge of nonintersecting member combinations to reduce storage requirement estimation in the design algorithm. For more information, see Dependent Dimensions.

Ragged Dimensions

Ragged dimensions have at least one member whose logical parent is not in the level immediately above the member, such as is the case of countries that do not have a state or province level between the country and city levels. For more information, see Ragged Dimension Support.

Enhanced Virtual Dimensions

Virtual dimensions have better performance and greater flexibility than in earlier versions. They can be based directly on columns in another dimension's table and include multiple levels. Virtual dimensions are no longer limited to 760 members. For more information, see Virtual Dimensions and Virtual Dimensions Created in Version 7.0.

Custom Rollup Formulas and Custom Member Formulas

Cube cell values associated with members can be calculated according to custom expressions rather than the aggregate functions of measures. The expressions can be applied to all the members in a level or individual members. For more information, see Custom Rollup Formulas and Custom Member Formulas.

Automatic Member Grouping

System-generated member groups can be used to accommodate members that have more than 64,000 children. This feature can also be used to provide an intermediate level for drilldown between a level with few members and one with numerous members. For more information, see Member Groups.

Default Members

Default members can be specified for each dimension and varied by role. They control end users' default views of cubes that include the dimension. For more information, see **Set Default Member** Dialog Box and Custom Rules in Dimension Security.

Dimension Filters

A filter (WHERE clause expression) can be used to limit the dimension table rows that are included in the dimension. The filter is specified in the **Source Table Filter** property of the dimension. For more information, see Properties Pane (Dimension Editor Data View).

Analysis Services

Data Mining Enhancements

Data mining technology analyzes data in relational databases and OLAP cubes to discover information of interest. The data mining features of Microsoft® SQL Server™ 2000 Analysis Services are incorporated in an open and extensible implementation of the new OLE DB for Data Mining specification. SQL Server 2000 includes data mining algorithms developed by Microsoft Research.

Relational and OLAP Data Mining

Analysis Services has incorporated data mining technology so that you can use it to discover information in relational databases and in OLAP cubes in Analysis Services. You can use the results of data mining to create a dimension that you can add to a cube to further analyze your data. For more information, see Data Mining Models.

Microsoft Decision Trees

The Microsoft Decision Trees algorithm uses classification techniques to analyze data. It then constructs one or more decision trees that can be used to predict attributes or values for new data. For example, you can use this algorithm to analyze credit history data and predict the credit risk of new applicants. For more information, see Microsoft Decision Trees.

Microsoft Clustering

The Microsoft Clustering algorithm uses a nearest neighbor method to group records into clusters that share similar characteristics. Often, these characteristics may be hidden or nonintuitive. For more information, see Microsoft Clustering.

Data Mining User Interface

Analysis Services provides new user interface wizards, dialog boxes, and editors to help you quickly perform data mining administrative tasks such as building data mining models and incorporating the results in OLAP cubes. You can browse a single decision tree model or the dependency network model of multiple trees produced when multiple attributes are predicted. For more information, see Building and Using Data Mining Models.

MDX Extensions for Data Mining

Multidimensional Expressions (MDX) syntax has been extended to provide data mining capabilities in connection with OLAP cubes. For more information, see MDX.

DTS Tasks for Data Mining

The Analysis Services Processing task has been enhanced to allow processing mining models, and a new Data Transformation Services (DTS) task is provided that you can use to create predictions from mining models. For more information, see Automating and Scheduling Administrative Tasks.

Data Mining in Client Applications

Client applications for Analysis Services can use data mining algorithms to discover information in OLAP cubes by creating data mining models and virtual cubes. For more information, see Advanced Data Mining and Analysis.

Analysis Services

Security Enhancements

This release of Microsoft® SQL Server™ 2000 Analysis Services includes features that provide more flexibility in controlling access to cube data, additional methods for authentication of users, and enhanced enforcement of roles.

Dimension Security

You can use roles to control end-user access to dimensions. For each role, you can limit access to individual dimensions, levels, and members, and you can set various read and read/write permissions. For more information, see Dimension Security.

Enhanced Cell Security

Analysis Manager now includes dialog boxes that enable you to define cell security by setting role options to control end user access to cube cells. You can limit a role's access to any combination of a cube's cells and you can vary the read and read/write permission of the role. For more information, see Cell Security.

Additional Authentication Methods

You can use HTTP authentication methods for client connections to Analysis servers. Analysis Services also supports Windows 2000® negotiated protocol authentication. For more information, see Server Security and Authentication.

Enhanced Role Enforcement

Role specifications are enforced on Analysis servers installed on either file allocation table (FAT) or NTFS file systems.

Analysis Services

Client Connectivity Enhancements in PivotTable Service

Enhancements to PivotTable® Service provide new connection options and support new cube and security features introduced in this release of Microsoft® SQL Server™ 2000 Analysis Services. Additional new functionality in PivotTable Service can be used by client applications when connected to an Analysis server or when working offline. For more information, see What's New in PivotTable Service.

Connecting to the Analysis Server

Client applications can communicate with the Analysis server through Microsoft Internet Information Services (IIS) using HTTP or HTTPS. Third-party security providers can also be used.

Allocated Writeback

You can write data to aggregation cells in a cube and, using one of four different allocation formulas, automatically distribute the data to the lowest level members. For more information, see Using Writebacks.

Managing Local Cubes

Many of the new features for server cubes, such as new dimension types, custom rollups, and calculated members, can also be used in local cubes.

Data Mining

PivotTable Service supports the advanced data mining and analysis techniques that are available in SQL Server 2000 Analysis Services. You can create and use data mining models to analyze data in server cubes or local databases.

Analysis Services

Other Enhancements

This release of Microsoft® SQL Server™ 2000 Analysis Services incorporates various enhancements to improve usability, processing performance, and MDX functionality.

Multiuser Administration

Multiple users can administer an Analysis server using Analysis Manager. Locking is applied only to the objects being edited and their dependent objects. For more information, see Analysis Manager.

Integration of Add-ins

The Archive and Restore Databases Add-in and the Copy and Paste Objects Add-in originally created for SQL Server version 7.0 OLAP Services are upgraded, installed, and fully integrated in this release. You can now use Analysis Manager to archive and restore SQL Server 2000 Analysis Services databases and to copy and paste objects in the Analysis Manager tree pane. For more information, see Archiving and Restoring Databases and Copying and Pasting Objects.

MDX Enhancements

A number of new functions have been added to the Multidimensional Expressions (MDX) syntax that you can use for calculated members and increased analytical evaluation. For more information, see MDX.

MDX Builder

A new MDX Builder tool helps you create MDX expressions using drag-and-drop techniques. The MDX editing text box has been enhanced to provide color coding of keywords and functions, indication of parentheses balance, and smart tips that show function syntax. For more information, see MDX Builder.

Table Aliases

In Cube Editor and Dimension Editor you can specify table aliases. For more information, see Schema Tab (Cube Editor Schema View) and Schema Tab (Dimension Editor Schema View).

Active Directory Support

Analysis Services supports Active Directory™, the directory service that is included with Microsoft Windows® 2000. You can enable an Analysis server so that it publishes Active Directory entries when the server starts and updates the entries when the server status changes. For more information, see Using Active Directory with Analysis Services.

Performance Monitor Counters

A number of performance counters are now available for Analysis server that can be used with Performance Monitor in Microsoft Windows NT® 4.0 or System Monitor in Windows 2000. For more information, see Monitoring Analysis Services Performance.

Analysis Services

Installing Analysis Services

This section contains information about installing Microsoft® SQL Server™ 2000 Analysis Services only. It does not contain information about installing other components of SQL Server 2000. For more information about installing other components, such as English Query, see Getting Started with SQL Server Books Online.

This section contains the following topics.

Topic	Description
Hardware and Software Requirements for Installing Analysis Services	Provides the hardware and software requirements for installing and running Analysis Services.
Running Setup	Provides step-by-step instructions to install Analysis Services.
Setup Parameters and Silent Installation	Describes the parameters for the Analysis Services Setup program (Setup.exe).
Reinstalling Analysis Services	Describes how to reinstall Analysis Services.
Removing Analysis Services	Describes how to remove Analysis Services.
Upgrading from an Earlier Version	Describes how to upgrade from an earlier version of Analysis Services.
Backward Compatibility	Provides information about compatibility with previous versions of Analysis Services (formerly called OLAP Services).

Related Documents

The Readme.html file in the root directory of the SQL Server 2000 CD-ROM contains information about Analysis Services. You can also view the release notes by clicking **Read the Release Notes** on the SQL Server 2000 Setup program (Autorun.exe) menu.

Analysis Services

Hardware and Software Requirements for Installing Analysis Services

Before you can install Microsoft® SQL Server 2000™ Analysis Services, your computer must meet the following requirements.

Hardware/software	Requirements
Computer	Intel® or compatible (Pentium 133 MHz or higher, Pentium PRO, Pentium II, or Pentium III)
Memory (RAM)	32 megabytes (MB) minimum (64 MB recommended)
Disk drive	CD-ROM drive
Hard disk space [(1)]	50 – 90 MB (130 MB for all components including common files and samples), 12 MB for the client only
Operating system	Microsoft Windows® 2000 Server [(3)] -or- Microsoft Windows NT® Server 4.0 with Service Pack 5 or later [(3)] For client components on client computers only, the following systems also qualify: Windows 2000 Professional Windows NT Workstation 4.0 with Service Pack 5 Windows 98 Windows 95 + DCOM95 Windows 95 OSR2 + DCOM95
Network software	Windows 2000, Windows NT 4.0, Windows 98, or Windows 95 built-in network software; and TCP/IP (included with Windows).
Online product documentation viewer	Microsoft Internet Explorer version 5.0 or later [(2)]. You must install Windows NT 4.0 Service Pack 5 or later before you install Internet Explorer version 5.0.

Access permissions	To install the services for Analysis server, you must be logged on to the server with Administrator permissions.

1 Setup installs a number of components that can be shared by other applications and may already exist on the computer.

2 Internet Explorer is required for Microsoft Management Console (MMC) and HTML Help. A minimal installation is sufficient, and Internet Explorer does not need to be your default browser. Internet Explorer is not required for the client-only installation.

3 Analysis Services should not be installed on a domain controller; this installation configuration is not supported.

For more information about supported hardware, see the Microsoft Windows Hardware Compatibility List at the Microsoft Web site. For more information about Windows 2000-compatible hardware, use the Microsoft Windows 2000 compatible hardware devices search tool at the Microsoft Web site.

Analysis Services

Running Setup

This topic describes how to install Microsoft® SQL Server™ 2000 Analysis Services.

If you are upgrading from an earlier version of Analysis Services (formerly called OLAP Services), you should take certain steps before performing the following procedure. For more information, see Upgrading from an Earlier Version.

If you are reinstalling Analysis Services, you should take certain steps before and after performing the following procedure. For more information, see Reinstalling Analysis Services.

Although Analysis Services can connect to multiple instances of SQL Server running on a single computer, you cannot install multiple instances of Analysis Services on a single computer.

To install Analysis Services, use the Analysis Services Setup program or the SQL Server 2000 Setup program.

To install Analysis Services

1. Exit all Microsoft Windows® applications.
2. Insert the SQL Server 2000 CD into the CD-ROM drive. This starts the SQL Server 2000 Setup program. If the Setup program does not start automatically, run the Autorun.exe program in the root directory of the CD-ROM.
3. Click **Install SQL Server 2000 Components**.
4. Click **Analysis Services** to start the Analysis Services Setup program.
5. In the **Welcome** step, click **Next**.

6. In the **Software License Agreement** step, read the license agreement, and then do one of the following:
    - To accept the license agreement, click **Yes**. You must select this option to install Analysis Services.
    - To reject the license agreement, click **No**. If you select this option, the program will ask you to confirm exiting. If you select **Exit Setup**, the program closes and the installation is canceled. To continue Setup, click **Resume**.
7. The Setup program prompts you to enter the CD key. Type the 10-digit CD key for the product, and then click **OK**.
8. The Setup program displays the complete product ID, which you can record for future reference. After you record the product ID, click **OK**.
9. In the **Select Components** step, select the components you want to install. All of the options are selected by default. You cannot clear the check box of any component on which another selected component depends.

    Unless you are installing the client components on a client computer, installing all components is recommended. The following components are available for installation.

Component	Description
Analysis server	Binary executables and other server-related files required for an installation of an Analysis server. Includes the **FoodMart 2000** sample database used by the tutorial. Requires the client components.
Analysis Manager	Binary executables and other files that support the user interface for administering the Analysis server. Includes the MDXSample executable

	file. Requires Decision Support Objects (DSO) and the client components.
Decision Support Objects	The object model for administering the Analysis server and managing meta data. Requires the client components.
Client components	Binary executables and related files for the Analysis Services client. Client components include PivotTable® Service.
Sample applications	Sample applications include the MDXSample source files, the **FoodMart 2000** database, and programming samples. Requires the client components.
Books Online	The entire documentation set for SQL Server 2000, including Analysis Services. This file is approximately 30 megabytes (MB). If space is at a premium, you can choose not to install Books Online. However, product documentation will not be available in the user interface until it is reinstalled.

To change the destination drive or folder, click **Browse**. Although remote network drives are listed in these dialog boxes, installation to locations on remote network drives is not supported.

Space Required and **Space Available** indicate disk drive space and help you determine what components to install. If your current disk drive does not have enough space available, you can click **Disk Space** to determine which disks on your computer have enough space to install Analysis Services.

After you select the components to install, click **Next**. The steps that follow may change depending on which components you selected to install.

10. In the **Data Folder Location** step, you can change the location of the

Data folder, which is the data storage location of the Analysis server.

The default location for the Data folder is C:\Program Files\Microsoft Analysis Services\Data (unless you specified another location for Analysis Services in the previous step). You can specify a different location by clicking **Browse**. If you change the default folder or drive, be sure to enter a fully qualified path. To specify a data storage location other than the computer on which the server is installed, you must have full control access permissions on that computer.

IMPORTANT The Data folder stores security files that control end users' access to objects on the Analysis server. For this reason, the Data folder must be secured against unauthorized access.

After you select the location of the Data folder, click **Next**.

11. In the **Select Program Folder** step, accept the default program folder name or enter a new one. This determines the location of the Analysis Services menu items on the **Start** menu. Click **Next**.
12. Analysis Services installation begins. After Setup notifies you that the installation is complete, click **Finish**.
13. If you are prompted to restart your computer, do one of the following:
    - Click **Yes, I want to restart my computer now**, and then click **Finish**.
    - Click **No, I will restart my computer later**, and then click **Finish**. If you select this option, the installation is not complete until after you restart the computer.
14. If you are finished installing SQL Server 2000 components, click **Exit** in the SQL Server 2000 Setup program.

If in Step 10 you specified a data storage location other than the computer on which the server is installed, you must configure your Analysis server service (MSSQLServerOLAPService) to log on as your user account, instead of the default, which is to log on as the system account. To do this, use the Services application, which is in Control Panel in Windows NT® 4.0 or the Administrative Tools folder in Control Panel in Windows 2000.

Analysis Services

Setup Parameters and Silent Installation

You can start the Analysis Services Setup program (\Msolap\Install\Setup.exe on the SQL Server CD-ROM) with the following optional command line parameters:

-r

> This option causes Setup.exe to automatically generate a silent response file (.iss), which is a record of the installation input, in the systemroot folder (typically C:\WinNT).

-s

> This option performs a silent (unattended) installation.

-f1<path\ResponseFile>

> This option allows you to specify the alternate location and name of the response file (.iss file). If the -f1 switch is not used when you run silent installation, Setup searches for the response file Setup.iss in the same folder as Setup.exe.

-f2<path\LogFile>

> This option allows you to specify an alternate location and name of the log file. By default, the Setup.log log file is created and stored in the systemroot folder (typically C:\Winnt).

If you use the **-r** option you can create a record of any installation scenario. You can use this record to perform a silent (unattended) installation. For example, the following command initiates a silent installation of the components specified in the Setup.iss response file previously recorded when you used the **-r** option:

```
Setup.exe -s -f1C:\temp\setup.iss
```

-z

> Prevents Setup.exe from checking the available memory during initialization. This switch is necessary when running Setup on a computer with more than 256 megabytes (MB) of memory. If it is not used, Setup.exe reports

insufficient memory and exits.

Analysis Services

Reinstalling Analysis Services

To reinstall Microsoft® SQL Server™ 2000 Analysis Services, follow these steps:

1. If you have made changes to the FoodMart 2000 sample database and want to preserve changes, back up FoodMart2000.mdb, which is installed by default to: C:\Program Files\Microsoft Analysis Services\Samples. Otherwise, this file is overwritten during the installation process.

2. Install Analysis Services. For more information, see Running Setup.

Note Reinstalling Analysis Services does not delete the Analysis Services repository (Msmdrep.mdb), which contains Analysis Services meta data. However, you must process all cubes in the repository after reinstallation. If you have backed up the **FoodMart 2000** sample database before reinstallation, restore FoodMart2000.mdb to recover your changes to the file.

Analysis Services

Stopping or Removing Analysis Services

To stop Microsoft® SQL Server™ 2000 Analysis Services, follow these steps:

1. Open Control Panel.

2. If your computer's operating system is Windows® 2000, open the Administrative Tools folder, and then double-click **Services**.

   If your computer's operating system is Windows NT® 4.0, double-click **Services**.

3. Select MSSQLServerOLAPService, and then on the **Action** menu click **Stop**.

4. Wait until the application notifies you that the service has stopped.

To remove Analysis Services, use the Add/Remove Programs application in Control Panel. Removing Analysis Services does not delete the Analysis Services repository (Msmdrep.mdb), which contains Analysis Services meta data, or the query log (Msmdqlog.mdb). If you want to fully remove Analysis Services, you must delete these files manually.

Analysis Services

Upgrading from an Earlier Version

To upgrade from an earlier version of Microsoft® SQL Server™ 2000 Analysis Services (previously OLAP Services), perform the following actions:

- Back up the Analysis Services repository and query log.

- Before you install Analysis Services, as a precaution against data loss, back up the Analysis Services repository (Msmdrep.mdb), which contains Analysis Services meta data, and the query log (Msmdqlog.mdb). These files are located in the Bin folder in the Analysis Services folder.Run Setup.

  Install Analysis Services by running the Analysis Services Setup program. For more information, see Running Setup.

When you upgrade from an earlier version, Setup does not delete or replace the Analysis Services repository or the query log.

Note The default location for Analysis Services has changed from C:\Program Files\OLAP Services in earlier versions of Analysis Services to C:\Program Files\Microsoft Analysis Services in this version of Analysis Services.

Analysis Services

Backward Compatibility

Microsoft® SQL Server™ 2000 Analysis Services is compatible with SQL Server version 7.0 OLAP Services. Cubes that were created in SQL Server 7.0 OLAP Services need to be migrated to the updated meta data repository format and reprocessed. Otherwise, the existing structures for cubes, roles, shared dimensions, and so on do not need to be changed. For more information about migrating the SQL Server 7.0 OLAP Services repository to SQL Server 2000 Meta Data Services, see Migrating Analysis Services Repositories.

The following sections concern backward compatibility with SQL Server 7.0 OLAP Services.

Administration of Analysis Services

Analysis Manager is backward compatible with SQL Server 7.0 OLAP Services. It is capable of administering both OLAP servers (the server that ships with SQL Server 7.0 OLAP Services), and Analysis servers (the server that ships with SQL Server 2000 Analysis Services) concurrently. When administering an OLAP server, the OLAP Services portion of SQL Server 7.0 Service Pack 2 code is used to assure complete backward compatibility. The add-in programs in Service Pack 2 are now integrated with Analysis Manager and do not need to be installed to administer OLAP servers.

Client and Local Cube Support

Some features in SQL Server 2000 Analysis Services are not supported by the SQL Server 7.0 OLAP Services client components or in a local cube. For more information, including a list of features, see 7.0 Analysis Services Client and Local Cube Support.

Decision Support Objects

Analysis Services now includes an updated version of Decision Support Objects (DSO), which is automatically installed during Setup. Programs must use this updated version of DSO when administering an Analysis server (the server that ships with SQL Server 2000 Analysis Services). No other change to these programs is necessary. Programs that use the updated version of DSO are compatible with and can administer OLAP servers (the server that ships with SQL Server 7.0 OLAP Services); however, new features will not be available on the OLAP server.

PivotTable Service

SQL Server 2000 Analysis Services includes an updated version of PivotTable® Service. Client applications that use PivotTable Service do not need to use this new version when connecting to an Analysis server unless you need access to objects that include new features. The objects that use these new features (such as data mining models and cubes that include parent-child dimensions) are not seen by the client applications that use the earlier version of PivotTable Service. Client applications that use the updated version of PivotTable Service can connect to any server, regardless of its version. Client applications that use the updated version of PivotTable Service can configure their compatibility settings using the following properties:

- **MDX Compatibility** property
- **MDX Unique Name Style** property
- **Secured Cell Value** property
- **Visual Mode** property

Custom Add-in Programs

Custom add-in programs that were developed for use with SQL Server 7.0 OLAP Services will continue to work with SQL Server 2000 Analysis Services. No changes are necessary to use them.

Archiving, Restoring, and Migrating Data

Analysis Services supports some but not all permutations of archiving and restoring databases and migrating repositories between versions of the product.

For information about supported migration paths, see Supported Migration Paths for Analysis Services Repositories. For information about archiving and restoring data between versions of the product, see Archiving and Restoring Databases Between Versions of Analysis Services.

Analysis Services

7.0 Analysis Services Client and Local Cube Support

This table shows support for new server features by the Microsoft® SQL Server™ 7.0 OLAP Services client components and in a SQL Server 2000 Analysis Services local cube. When a feature may cause data to be translated incorrectly by a 7.0 client application, the server prevents the cube from being visible and prevents the client connection to the cube. If the absence of a feature in a local cube might change data values presented to the user, then a local cube using the feature cannot be created.

For each feature listed here, the table shows whether a cube containing a feature is visible on a 7.0 client application and if the cube is visible whether the feature itself is available on the 7.0 client application. For each feature, the table also shows whether a local cube can be created using the feature and whether the feature itself is supported in a local cube.

Feature	**Cube is visible on 7.0 client**	**Feature available on 7.0 client**	**Can create local cube using feature**	**Supported in a local cube**
Actions	Yes	No	Yes	No
Additional authentication methods	Yes	Yes	Yes	(2)
Calculated cells	No	No	No	No
Changing dimensions	Yes	No	Yes	No
Custom member formulas	No	No	No	No
Custom rollup formulas	No	No	No	No
Default members	No	No	Yes	Yes
Dimension security	No	No	No	No
DistinctCount	No	No	No	No
Drillthrough	Yes	No	Yes	No
Enhanced cell Security	Yes	Yes	Yes	No

Enhanced virtual dimensions(1)	Yes	Yes	Yes	Not applicable
Exceeding 7.0 Limits(3)	No	No	Yes	Yes
Linked cubes	Yes	Not applicable	Yes	No
Member groups	Yes	Yes	Yes	Yes
Members with data	Yes	Yes	Yes	Yes
New MDX functions	Yes	No	Yes	(4)
Parent-child dimensions	No	No	Yes	Yes
Ragged dimensions	Yes	Yes	Yes	Yes
ROLAP dimensions	Yes	Not applicable	Yes	No
Siblings with same names	No	No	Yes	Yes
Write-enabled dimensions	Yes	No	Yes	No

1 The earlier limit of 760 members in a virtual dimension does not apply.
2 Cell security is not supported on local cubes.
3 Exceeding 127 measures in a cube, 63 dimensions in a cube, or 128 levels in a cube. For information about SQL Server 2000 Analysis Services limits, see Specifications and Limits.
4 For the SQL Server 7.0 OLAP Services client, new Multidimensional Expressions (MDX) functions are not supported. For local cubes, new MDX functions are available, except for **LookUpCube**. Calculated members using **LookUpCube** in local cubes are not created.

Analysis Services

Supported Migration Paths for Analysis Services Repositories

You can migrate a Microsoft® SQL Server™ 2000 Analysis Services repository from the default Microsoft Access (Microsoft Jet 3.5 or 4.0) database to a SQL Server database on the same or a different computer. You cannot migrate a SQL Server repository to a Microsoft Access repository. You can change the format from native to SQL Server 2000 Meta Data Services format when you migrate a database. To migrate a SQL Server database repository between native and Meta Data Services formats, you must migrate it from one SQL Server database to another. The following table shows supported migration paths for repository databases.

		To native		**To MDS**
		Jet 3.5/4.0	**SQL Server 7.0/2000**	**SQL Server 2000**
From native	**Jet 3.5/4.0**	No	Yes	Yes
	SQL Server 7.0/2000	No	Yes(2)	Yes(2)
From MDS(1)	**SQL Server 2000**	No	Yes(2)	Yes(2)

1 MDS represents the Meta Data Services (previously named Microsoft Repository) format supported by SQL Server 2000.
2 Source and destination must be different databases.

See Also

Migrating Analysis Services Repositories

OLE DB Provider for Jet

Analysis Services

Archiving and Restoring Databases Between Versions of Analysis Services

On an Analysis server (the server that ships with Microsoft SQL Server 2000 Analysis Services), you can restore databases that were archived using an OLAP server (the server that ships with SQL Server 7.0 OLAP Services) or an Analysis server. The following table shows all the restoration paths supported for databases archived while in SQL Server 7.0 OLAP Services or SQL Server 2000 Analysis Services using native or SQL Server 2000 Meta Data Services formats with SQL Server or the Microsoft Jet 3.5 or 4.0 OLE DB provider.

			To native				**To MDS**
			Jet 3.5/4.0		**SQL Server**		**SQL Server**
			7.0	**2000**	**7.0**	**2000**	**2000**
From native[1]	**Jet 3.5/4.0**	**7.0**	Yes	Yes	Yes	Yes	Yes
		2000	No[3]	Yes	No[3]	Yes	Yes
	SQL Server	**7.0**	Yes	Yes	Yes	Yes	Yes
		2000	No[3]	Yes	No[3]	Yes	Yes
From MDS[2]	**SQL Server**	**2000**	No[3]	Yes	No[3]	Yes	Yes

1 **From** specifies the repository format, database engine, and version of OLAP Services or Analysis Services that archives a database; **To** specifies the repository format, database engine, and version of OLAP Services or Analysis Services that restores a database.
2 MDS represents the Meta Data Services (previously named Microsoft Repository) format supported by SQL Server 2000.
3 OLAP servers do not support restoration of Analysis Services databases.

See Also

Archiving and Restoring Databases

OLE DB Provider for Jet

Analysis Services

Analysis Services Architecture

The following topics contain information about the architecture of Microsoft® SQL Server™ 2000 Analysis Services.

Topic	Description
Analysis Services Features	Contains an overview of Analysis Services and its main features.
Server and Client Architecture	Describes the interaction between the Analysis server and client applications.
Object Architecture	Contains information about the objects you work with in Analysis Services and how they are connected.
Security and Authentication	Describes security in Analysis Services, which bases roles on Windows NT® 4.0 and Windows® 2000 users and groups to provide security settings that you can set anywhere from the database level to the individual cell level.

Analysis Services

Analysis Services Features

Microsoft® SQL Server™ 2000 Analysis Services is a middle-tier server for online analytical processing (OLAP) and data mining. The Analysis Services system includes a server that manages multidimensional cubes of data for analysis and provides rapid client access to cube information. Analysis Services organizes data from a data warehouse into cubes with precalculated aggregation data to provide rapid answers to complex analytical queries. Analysis Services also allows you to create data mining models from both multidimensional (OLAP) and relational data sources. You can apply data mining models to both types of data. PivotTable® Service, the included OLE DB compliant provider, is used by Microsoft Excel and applications from other vendors to retrieve data from the server and present it to the user, or create local data cubes for offline analysis.

Certain features are available only if you install Analysis Services for certain editions of SQL Server 2000. For more information about which editions support which features, see Features Supported by the Editions of SQL Server 2000.

The following table describes the key features of Analysis Services.

Topic	Description
Ease of Use	An extensive user interface with wizards
Flexible Data Model	A flexible, robust data model for cube definition and storage
Scalability	Scalable architecture that provides a variety of storage scenarios and an automated solution to the data explosion syndrome that plagues traditional OLAP technologies
Integration	Integration of administration tools, security, data sources, and client/server caching
Widely Supported APIs and Open Architecture	Support for custom applications

Analysis Services

Ease of Use

To make online analytical processing (OLAP) and data mining technology easier to use, Microsoft® SQL Server™ 2000 Analysis Services provides wizards, editors, tools, and information within Analysis Manager. This console application provides a user interface for accessing Analysis servers and their meta data repositories.

Tutorial and Overview Material

You can use the online tutorial to master Analysis Manager in a few hours. Designed for both beginners and experienced OLAP users, the tutorial walks you through the steps for creating a basic cube, as well as more advanced operations, such as creating partitions, virtual cubes, security roles, writable dimensions, actions, and data mining models. The tutorial is an excellent tool for learning about OLAP, data mining, and the operation and features of Analysis Manager.

You can also find information about OLAP and Analysis Services in the HTML pane (the right pane) of Analysis Manager.

Meta Data and Data View

In the right pane of Analysis Manager, you can view object properties and meta data and browse data for cubes and data mining models as you traverse the tree pane.

Cube Wizard

With this easy-to-use wizard, you can build all the structures necessary to create an OLAP cube. The wizard walks you through the entire cube design and implementation process, from mapping data sources and creating dimensions to defining measures.

Cube Editor

Using simple drag-and-drop operations, you can edit existing cube structures and create new ones. Cube Editor complements the Cube Wizard. Using Cube

Editor, you can revise cubes you created with the wizard or quickly create new ones.

Dimension Wizard

Using the Dimension Wizard, you can quickly and easily create a shared dimension, which can be used by any cube, or a private dimension, which can be used in a single cube. You can map database dimension table columns to dimension levels or use the built-in time dimension generator to create a variety of time dimensions based on a date-time column in the database. You can use the Dimension Wizard to create dimensions based on star or snowflake data warehouse schemas. In addition, you can also create parent-child, virtual, and data mining dimensions.

Dimension Editor

Using simple drag-and-drop operations, you can edit existing shared dimension structures and create new ones. Dimension Editor complements the Dimension Wizard, enabling you to revise dimensions you created with the wizard or to quickly create new ones. You can also preview dimension data in the editor.

Incremental Update Wizard

You can use this wizard to guide you through the process of incorporating new data into your cube. An incremental update adds new data to a cube without the necessity of rebuilding aggregations and reloading all data.

Partition Wizard

This wizard helps you create a new partition to contain a portion of the data in your cube. Partitions enable you to distribute and optimize a cube's data into discrete segments on a single server or across multiple servers.

Note You can create multiple partitions in a cube only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Storage Design Wizard

You can use this wizard to specify the storage mode for cube data and to help you design aggregations appropriate to the intended use of your cube. The options available in this wizard enable you to optimize the tradeoff between response time and storage requirements according to the needs of your application and users.

Usage Analysis Wizard

By displaying logged query information such as date, user, query response time, and frequency in tabular and graphical format, the Usage Analysis Wizard helps you understand how a cube is being used.

Usage-Based Optimization Wizard

The Usage-Based Optimization Wizard helps you tune cube performance based on users' actual usage of the cube. You can direct the wizard to create aggregations to improve performance based on any combination of users, the number of times a query was executed, query response time, the mode of storage where the data resides, or a date range.

Calculated Cells Wizard

This wizard helps you create calculated cells, by allowing you to define a subsection of a cube, referred to as a subcube, whose value is determined by a Multidimensional Expressions (MDX) formula. The functionality of calculated cells is similar to that of custom members, except that calculated cells can affect specific cells, even a single cell, within a cube, allowing finer control for financial and statistical calculations.

Action Wizard

You can use this wizard to create an action associated with a cube or a portion of a cube. An action allows users to trigger an operation on a selected cube or a part of a cube and automatically pass the selected item as a parameter to the operation. For example, a user can select an action on a dimension member that automatically opens his or her Internet browser so he or she can access a page about the member.

Virtual Cube Wizard

You can use this wizard to join cubes and to select dimensions and measures from them to create a virtual cube. A virtual cube enables a single query to be routed to multiple cubes, including cubes running on different servers. A virtual cube appears to users as a regular cube, but it does not require additional storage space; it is similar to a view that joins tables in a relational database.

Virtual Cube Editor

Using simple drag-and-drop operations, you can edit existing virtual cube structures. Virtual Cube Editor complements the Virtual Cube Wizard, enabling you to revise virtual cubes you created with the wizard.

Mining Model Wizard

You can use this wizard to create data mining models from both OLAP and relational data sources. You can also specify different data mining techniques to build your model. If you are creating a mining model based on OLAP data, you have the option to create a dimension and virtual cube to help you analyze the mining model results.

Mining Model Editors

In these two editors, one for OLAP data mining models and the other for relational data mining models, you can edit existing mining models using drag-and-drop techniques and browse the results of your mining models.

Data Views

You can view data for cubes, dimensions, and data mining models without leaving Analysis Manager. You do not need to switch to another application to check your designs.

OLE DB Data Source Locator Integration

Analysis Services uses the Microsoft Data Source Locator component for selecting OLE DB or ODBC data sources.

Role Managers

Using Database Role Manager and Cube Role Manager, you can create and maintain roles to control users' access to cubes and their component parts.

Analysis Services

Flexible Data Model

By supporting various data and storage models, Microsoft® SQL Server™ 2000 Analysis Services helps you create and maintain a system that meets your organization's needs.

Multiple Data Storage Options

Analysis Services offers three storage modes for dimensions, partitions, and cubes:

- Multidimensional OLAP (MOLAP)

  The underlying data for a cube is stored along with aggregation data in a high-performance multidimensional structure. MOLAP storage provides excellent performance and data compression.

- Relational OLAP (ROLAP)

  The underlying data for a cube is stored along with the aggregation data in a relational database. ROLAP storage enables you to take advantage of your investment in relational technology and enterprise data management tools.

- Hybrid OLAP (HOLAP)

  The underlying data for a cube is stored in a relational database and the aggregation data is stored in a high-performance multidimensional structure. HOLAP storage offers the benefits of MOLAP for aggregations without necessitating duplication of the underlying detail data.

Note Dimensions with more than 10 million members must use the ROLAP storage mode. This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Partitioned Cube Storage

You can partition a cube into separate physical sections. Each partition can be stored in a different mode, in a different physical location, and with a level of aggregations appropriate to the data in the partition. The result is that you can fine-tune the performance and data management characteristics of your system.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Partition Merging

You can combine a cube's multiple partitions back into a single physical partition. For example, you can use partition merging to consolidate portions of cube data, such as data for a just completed quarter into a single partition for the year.

Write-Enabled Cubes

You can enable a cube for write access by multiple simultaneous users. User-initiated changes to the cube data are logged to a special, physically separated partition table associated with the cube and applied automatically as cube data is viewed. To the user it appears as if the data in the cube has changed. The changes can be discarded or made read-only at the discretion of the database administrator (DBA).

Balanced, Unbalanced, and Ragged Hierarchies

You can create dimensions with balanced or unbalanced hierarchies. Dimensions with balanced hierarchies have all branches of the hierarchy end at the same level while branches of unbalanced hierarchies terminate at different levels.

Ragged hierarchies are also supported. This allows dimensions in which at least one member does not have its logical parent in the level immediately above the member to accommodate levels where no values exist; the logical parent of a member in a ragged hierarchy can be two levels above the member.

Parent-Child Dimensions

You can create a dimension based on two dimension table columns that together define parent-child relationships between rows in the dimension table. Parent-child dimensions support balanced, unbalanced, and ragged hierarchies; complex hierarchical relationships can be easily created using parent-child dimensions.

Write-Enabled Dimensions

You can enable a dimension for write access by multiple simultaneous users. User-initiated changes to the dimension data are recorded in the dimension table. Users can manipulate the dimension data to see the immediate effect on the cube data.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Virtual Cubes

You can join cubes into virtual cubes, much like tables can be joined with views in a relational database. A virtual cube provides access to data in the combined cubes without necessitating the construction of a new cube, while it allows you to maintain the best design for each individual cube.

Calculated Members

You can create calculated measures and calculated dimension members by combining Multidimensional Expressions (MDX), mathematical formulas, and user-defined functions. This facility enables you to define new measures and dimension members based on a rich yet easy-to-use expression syntax. You can register additional libraries of user-defined functions to use in calculated member definitions.

Custom Unary Operators

Custom unary operators use simple math operators, called *unary operators*, stored in a column to determine how the value of a level member affects the value of the parent. Custom rollup operators are unique per level member.

Custom Rollup Formulas and Custom Member Formulas

Custom rollup formulas and custom member formulas are MDX expressions that determine cube cell values associated with members. Custom rollup formulas apply to all members of a level, whereas custom member formulas apply to individual level members.

Calculated Cells

Similar to custom member formulas, calculated cells are MDX statements that determine cube cell values associated with a specified group of cells. Calculated cells apply only to specified cells in a cube, whereas custom member formulas must apply to all of the cells for a given member.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Member Properties

You can define properties for dimension members and use data for these properties within a cube. For example, if the members of a Product dimension are SKUs, there are likely to be several properties associated with SKUs, such as size, color, fabric, and so on. You can specify such properties as member properties and use them in analytical queries.

Virtual Dimensions

A virtual dimension can be created from member properties or levels of another dimension. A virtual dimension can be used to evaluate the properties of a dimension's members against the members themselves. For example, measures can be evaluated for SKUs against size, color, fabric, and so on. Virtual dimensions and member properties are evaluated as necessary for queries and require no physical cube storage.

Analysis Services

Scalability

Microsoft® SQL Server™ 2000 Analysis Services provides a scalable architecture to address a variety of data warehousing scenarios.

Customized Aggregation Options

Using the Storage Design Wizard, you can optimize the tradeoff between system performance and the disk space allocated to storing aggregations. Analysis Services uses a sophisticated algorithm to determine the optimum set of aggregations from which other aggregations can be derived. As a result, you can focus on application design issues and leave the complex management of aggregation design up to the system.

Usage-Based Optimization

You can tune the performance of a cube to provide quick response to the queries most often executed by directing the Usage-Based Optimization Wizard to design aggregations appropriate to those queries while maintaining reasonable storage requirements. Thus, you can quickly build a system with a minimum number of aggregations and then later optimize performance according to the actual usage of the system.

Data Compression and Storage Optimization

In multidimensional OLAP (MOLAP) and hybrid OLAP (HOLAP) storage modes, Analysis Services stores all or some of the cube information in multidimensional structures. In these structures, storage is not used for empty cells, and a sophisticated data compression algorithm is applied to data that is stored. When combined with the flexible options for the design and optimization of precalculated aggregations, these techniques help to minimize the impact of the data explosion syndrome inherent in OLAP technology.

Distributed Calculation

PivotTable® Service incorporates functionality from the server so that calculations can often be performed on the client instead of the server. Because this distributes the computational load between the server and the client, it increases the capacity of the server, reduces network traffic, and improves performance for the clients.

Partitions

You can spread a cube over multiple servers by dividing it into partitions. Analysis Services can then retrieve data in parallel to answer queries. Partitioning enables you to manage your storage strategy, increase scale with multiple servers, and increase performance.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Linked Cubes

A cube can be stored on a single server and referenced as a linked cube on other servers. Users connected to any of these servers can then access the cube. This approach avoids the more costly alternative of storing and maintaining copies of a cube on multiple servers. Linked cubes make it possible for you to create, store, and maintain a cube on one Analysis server while providing access to the cube from multiple Analysis servers. Linked cubes facilitate cube security and reduce storage and maintenance requirements.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Distributed Partitioned Cubes

Creating distributed partitioned cubes by using remote partitions enables you to manage your storage strategy by storing a cube's data across multiple servers. Benefits include centralized administration, greater scalability, and enhanced performance through parallel processing of queries.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Incremental Updates

A cube can be updated by processing only the data that has been added rather than the entire cube; you can incrementally update OLAP cubes while they are in use.

LAN, WAN, Internet, and Mobile Scenarios

Intelligent cache management integrates the Analysis server with the PivotTable Service client, minimizing traffic over LAN and WAN connections. PivotTable Service contains an efficient multidimensional calculation engine to further minimize network traffic and to enable analysis of local multidimensional data when the client is not connected to the server.

Microsoft ActiveX® controls, Active Server Pages (ASP) scripting, and ActiveX Data Objects (ADO) APIs provide a variety of solutions for querying OLAP data over the Web.

To support mobile scenarios, you can create local cubes, which can be used without a connection to an Analysis server. Depending on their storage mode, some local cubes can be used without a connection to an Analysis server and without a connection to the cube's data source.

HTTP Authentication of Connections

HTTP or secure HTTP (HTTPS) authentication can be used in conjunction with Microsoft Internet Information Services (IIS) to establish connections to an Analysis server.

Note This feature is available only if you install Analysis Services for SQL Server 2000 Enterprise Edition.

Client Support for Windows 95 and Windows 98

PivotTable Service runs on Microsoft Windows® 95 and Windows 98, supporting client applications available for these platforms as well as for Microsoft Windows NT® 4.0 and Windows 2000.

Analysis Services

Integration

Microsoft® SQL Server™ 2000 Analysis Services works with other components and programs to ensure enterprise-level robustness.

Integrated Management Console

Analysis Services includes Analysis Manager, a graphical administration tool. This is a snap-in to Microsoft Management Console (MMC). It provides a common framework and user interface for defining, accessing, and managing Analysis servers and databases. You can use multiple snap-in components in MMC; for example, you can install SQL Server Enterprise Manager and other snap-in components along with Analysis Manager.

Integrated Security

Cube and data mining model access is based on Microsoft Windows NT® 4.0 or Windows® 2000 security, providing integration with SQL Server 2000 or earlier by way of operating system-level user account and group definitions.

OLE DB and ODBC Data Sources

A variety of OLE DB and ODBC data sources can be used, such as Oracle versions 7.3 and 8.0. You can use multiple sources at the same time.

Data Transformation Services

Using the Data Transformation Services (DTS) portion of SQL Server Enterprise Manager, you can create packages that process cubes and data mining models and run prediction queries based on mining models. These packages can be scheduled to execute automatically.

Meta Data Services

You can use SQL Server 2000 Meta Data Services to store Analysis Services meta data. Meta Data Services provides a store for object definitions and a platform for deploying meta data.

Functions from Excel and Visual Basic for Applications

In Multidimensional Expressions (MDX), you can include many functions in the Microsoft Excel worksheet library, which is automatically registered if installed on the computer with Analysis Services. You can also include many functions in the Microsoft Visual Basic® for Applications Expression Services library, which is included with Analysis Services and automatically registered.

Server-Side Cache

User queries, meta data, and data are stored in the Analysis server cache. Cached query definitions and meta data make it possible to answer new queries by calculating answers from cached data rather than retrieving data from the disk.

Client-Side Cache

Client applications connect to the Analysis server through the client-based PivotTable® Service component. Because PivotTable Service receives meta data with data from the server in response to a query, it can often use data in the client cache to calculate the answer to subsequent queries without sending a new query to the server. For example, if the client cache contains values for the four quarters of a specific year and the user asks for the total for the same year, PivotTable Service calculates the answer from the cached data.

PivotTable Service shares much of the same functionality as the server, enabling it to bring the server's multidimensional calculation engine, caching features, and query management directly to the client computer. This client/server data management model optimizes performance and minimizes network traffic.

Analysis Services

Widely Supported APIs and Open Architecture

Microsoft® SQL Server™ 2000 Analysis Services provides various tools you can use to programmatically extend its functionality.

OLE DB

Analysis Services is designed to meet the OLAP-specific requirements of the OLE DB 2.0 and later specification, which was developed with the input of over 40 OLAP client and server vendors and is a widely supported standard for multidimensional data access.

Analysis Services is also designed to meet the requirements of the OLE DB for Data Mining specification, which addresses data mining-specific provisions.

ADO

Analysis Services is compatible with Microsoft ActiveX® Data Objects (ADO) and its extension for multidimensional objects, ADO (Multidimensional) (ADO MD).

User-Defined Functions

You can extend the list of built-in functions by creating libraries of functions using Component Object Model (COM) automation languages, such as Microsoft Visual Basic® or Microsoft Visual C++®. You can register these libraries and use your functions in calculated member definitions and other expressions written in Multidimensional Expressions (MDX). This architecture enables you to add customized analysis tools.

Decision Support Objects

You can use the server object model, Decision Support Objects (DSO), to create applications that define and manage cubes, data mining models, and other objects. This object model can be used to extend the functionality of Analysis Manager or to automate the ongoing maintenance of your system.

Add-in Support

You can use the Analysis Services Add-in Manager interface to create applications that extend the functionality of the Analysis Manager user interface. Using the Analysis Services Add-in Manager interface and DSO, you can create custom extensions, dialog boxes, wizards, and other applications that integrate with Analysis Manager.

Analysis Services

Server and Client Architecture

Rapid access to data warehouse data is provided by Microsoft® SQL Server™ 2000 Analysis Services. Data from the data warehouse is extracted, summarized, organized, and stored in multidimensional structures for rapid response to end user queries.

Analysis Services also provides an architecture for access to data mining data. This data can be sent to the client in either a multidimensional or relational form.

Analysis Services and PivotTable® Service provide the capability to design, create, and manage cubes and data mining models from data warehouses and to provide client access to OLAP data and data mining data. The Analysis server manages the data; PivotTable Service works with the server to provide client access to the data.

Analysis Services

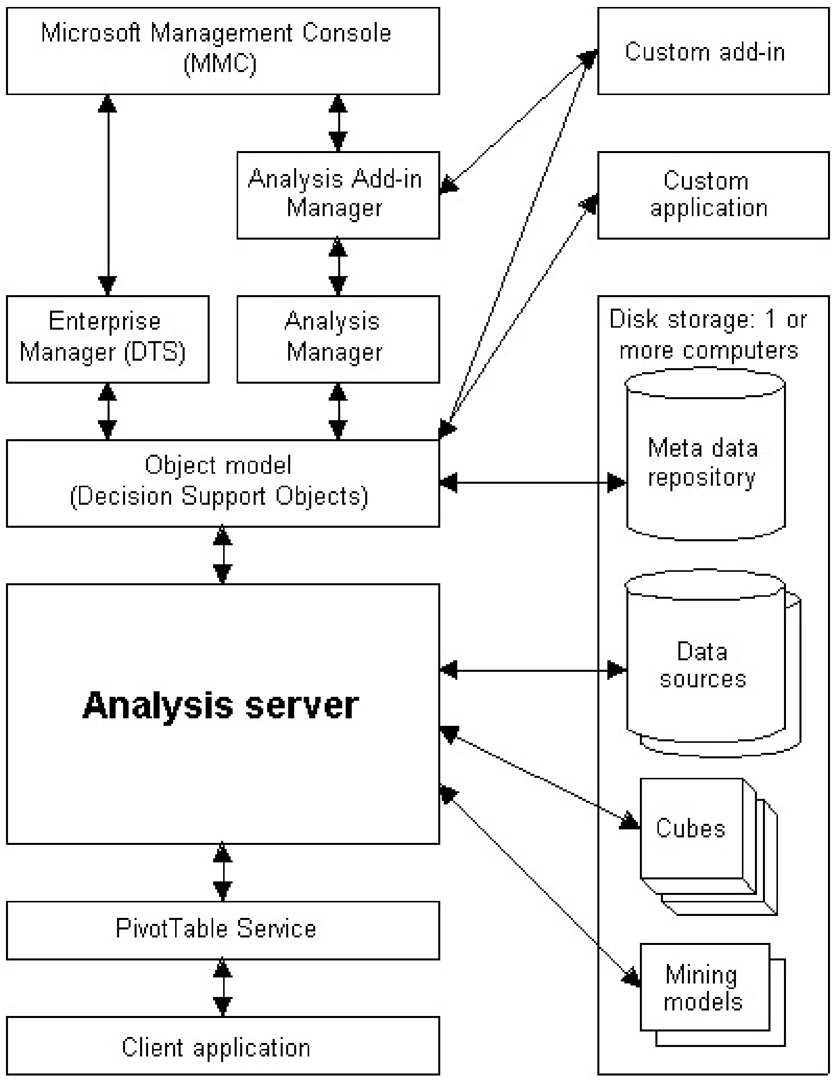
Server Architecture

Microsoft® SQL Server™ 2000 Analysis Services provides server capabilities to create and manage OLAP cubes and data mining models, and to provide this data to clients through PivotTable® Service. Server operations include:

- Creating and processing cubes from relational databases, usually in data warehouses.
- Storing cube data in multidimensional structures, in relational databases, or in combinations of both.
- Creating data mining models from cubes or from relational databases, usually in data warehouses.
- Storing data for data mining models in multidimensional structures, relational databases, or in Predictive Model Markup Language (PMML), which is a standardized XML format.

Meta data, the information used to define cubes, data mining models, and other objects on the server, is stored by Meta Data Services in a relational database.

A user interface is provided by the Analysis Manager add-in that runs under a snap-in to Microsoft Management Console (MMC). In addition, the Data Transformation Services (DTS) portion of SQL Server Enterprise Manager, which also snaps into MMC, provides a user interface with the server. Programming interfaces are provided to enable custom applications to interact with the object model that controls the server, as well as with Analysis Manager.



For more information about creating custom programs, see Programming Analysis Services Applications.

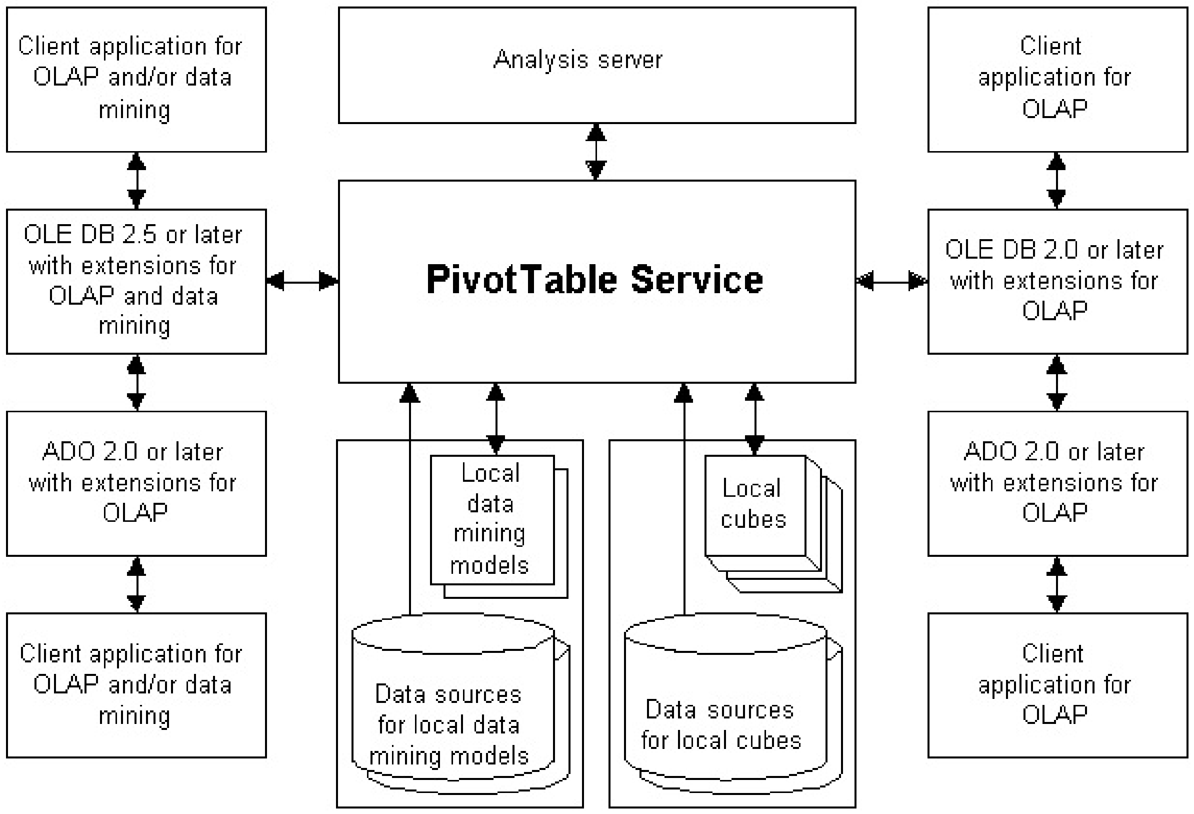
Analysis Services

Client Architecture

PivotTable® Service communicates with the Analysis server and provides interfaces that client applications can use to access OLAP data and data mining data on the server. Client applications connect to PivotTable Service using OLE DB interfaces for C++ or the Microsoft® ActiveX® Data Objects (ADO) object model for Component Object Model (COM) automation languages such as Microsoft Visual Basic®.

PivotTable Service can also create local cube files that contain data from a cube on the server or from OLE DB relational databases. Local cubes can be stored as multidimensional cube files on the client computer. Local cubes can be used offline with PivotTable Service for portable analysis. That is, to query local cubes, a connection to the Analysis server is not required; connection to the local cubes' data sources is not required if the local cubes have a multidimensional OLAP (MOLAP) storage mode.

PivotTable Service can also create local data mining model files that contain models processed from cubes or tables on the server or from OLE DB relational databases. Local data mining models can be stored on the client computer. Local data mining models can be used offline with PivotTable Service for portable analysis. That is, to use local data mining models, a connection to the Analysis server is not required.


Client application for OLAP and/or data mining
Analysis server
Client application for OLAP
OLE DB 2.5 or later with extensions for OLAP and data mining
PivotTable Service
OLE DB 2.0 or later with extensions for OLAP
ADO 2.0 or later with extensions for OLAP
Local data mining models
Local cubes
ADO 2.0 or later with extensions for OLAP
Client application for OLAP and/or data mining
Data sources for local data mining models
Data sources for local cubes
Client application for OLAP

For more information about programming the client, see PivotTable Service.

See Also

Key Concepts in PivotTable Service

Analysis Services

Object Architecture

The following table summarizes the objects used to administer Microsoft® SQL Server™ 2000 Analysis Services.

Topic	Description
Analysis Server	The server component of Analysis Services, designed specifically to create and maintain multidimensional data structures and to provide multidimensional data in response to client queries.
Databases	Databases serve as containers for related data sources, cubes, dimensions, data mining models, and the objects they share.
Data Sources	Data sources store the specification of the information necessary to access source data for an object such as a cube.
Dimensions	Dimensions serve as a structural attribute of a cube. A dimension is an organized hierarchy of categories (levels) that describe data in the fact table. These categories describe similar sets of members upon which the user wants to base an analysis.
Levels and Members	Levels are used as elements of a dimension hierarchy. Levels describe the hierarchy from the highest (most summarized) level to the lowest (most detailed) level of data.
Measures	Measures are, in a cube, a set of values that are based on a column in the cube's fact table and are usually

	numeric. In a cube, measures are the central values that are analyzed.
Cubes	Cubes contain a set of data that is usually constructed from a subset of a data warehouse and is organized and summarized into a multidimensional structure defined by a set of dimensions and measures.
Partitions	Partitions are the storage containers for data and aggregations of a cube.
Aggregations	Aggregations are defined as a table or structure containing precalculated data for a cube.
Roles	Roles contain a set of Microsoft® Windows NT® 4.0 or Windows® 2000 user accounts and groups with the same access to Analysis Services data.
Commands	Commands are used to hold an administrator-defined command that is automatically executed when a client accesses a database, cube, or role. Commands include calculated members, named sets, and actions.
Member Properties	Member properties contain information about the members of a dimension level in addition to that contained in the dimension.
Data Mining Models	Data mining models contain a virtual structure that represents the grouping and predictive analysis of relational or online analytical processing (OLAP) data.
Data Mining Columns	Data mining columns contain a structure that is used to define the content of a data mining model. A

	column can contain data or nested columns.

Analysis Services

Object Architecture Overview

Microsoft® SQL Server™ 2000 Analysis Services provides a variety of objects to help you implement an online analytical processing (OLAP) or data mining solution with a data warehouse. This topic includes descriptions of the available objects.

The main object of OLAP is the cube, which contains the current analytical data of interest to end users. To support the questions that end users ask, cubes organize data into dimensions and measures in a multidimensional structure. For example, consider the question, "What was our total sales of hardware in the northwest region in the first quarter of this year?" A cube of data that can answer this question includes three dimensions and one measure:

- The Product dimension, which contains a hardware category
- The Geography dimension, which contains the northwest region
- The Time dimension, which contains the first quarter of this year
- The Sales measure, which contains quantitative numerical data that can be summarized

Whereas OLAP allows you to perform aggregation analysis on current or past data, data mining actually allows prediction analysis to be performed based on current or past data. Instead of considering the question posed earlier in this topic using OLAP, the question, "What will our projected total sales of hardware in the northwest region be for the first quarter of next year?" can be asked and answered with data mining. The main object of data mining, the data mining model, provides a framework to store learned knowledge from your data, such as probability and distribution information, created from existing data to predict the behavior of new data. This, in turn, can be given new data for analysis, to predict expected values for a given case based on patterns and rules discovered in past data.

Object Hierarchy Diagram

The objects used to support OLAP and data mining are represented by a object hierarchy, used to maintain the complex relationships between the various objects, such as cubes, dimensions, and data mining models, that define Analysis Services.

The following diagram shows the positions of the objects within the Analysis Services object hierarchy. Some objects appear in multiple places within the hierarchy.

This topic describes the administrator's view of the object model. The programmer's view is somewhat broader and more complex. For more information, see Decision Support Objects.

Analysis Services

Analysis Server

The Analysis server is the server component of Microsoft® SQL Server™ 2000 Analysis Services. It is specifically designed to create and maintain multidimensional data structures and provide multidimensional data in response to client queries.

When you run Setup to install Analysis Services, you can choose to install the Analysis server component. The name of the Analysis server matches the computer name on the network. To connect to the Analysis server, client applications must specify this name in the connection string. End users typically supply a name to indicate the Analysis server they want to connect to.

Installing the Analysis server also creates the Analysis server object.

The Analysis server is the root object in the object hierarchy. As the root object, the Analysis server is the first object to be created and the object to which all other Analysis Services objects are subordinate. After an Analysis server is created, the next objects to be created are databases.

The principal tool for administering the Analysis server and its subordinate objects is Analysis Manager, which provides an extensive user interface. For more information, see Analysis Manager.

In Analysis Manager, a connected Analysis server is identified by the following icon.

Each Analysis server has a repository called the Analysis Services repository. This repository stores the meta data (that is, definitions) of the objects defined on the Analysis server. By default, the Analysis Services repository is Msmdrep.mdb on the computer running the Analysis server. However, the Analysis Services repository can be migrated to a SQL Server database. For more information, see Migrating Analysis Services Repositories.

Each Analysis server has an associated Data folder, which stores multidimensional structures for the objects defined on the Analysis server. These structures are contained in files that are created when the objects are processed. These structures are referenced to resolve queries sent to the Analysis server.

Some of these structures contain aggregations.

Important The Data folder also contains security files that control end users' access to objects on the Analysis server. For this reason, the Data folder must be secured against unauthorized access.

The location of the Data folder is specified during installation and can be changed. You can read and change the location of the Data folder in the **Properties** dialog box. You can also set other properties of the Analysis server in the **Properties** dialog box.

The service associated with the Analysis server is MSSQLServerOLAPService. By default, this service starts automatically and logs on as the system account. You can maintain the MSSQLServerOLAPService service in the Services application, which is either in Control Panel in Microsoft Windows NT® 4.0 or in the Administrative Tools folder in Windows® 2000.

If you are programming with Decision Support Objects (DSO), the class type associated with the Analysis server is **clsServer**.

See Also

Aggregations

clsServer

Configuring Analysis Servers

Databases

Dimension Processing

Operational Considerations

Processing Cubes

General Tab (Properties Dialog Box)

Running Setup

Server Security and Authentication

Analysis Services

Databases

A database is a container for related cubes and the objects they share. These objects include data sources, shared dimensions, and database roles. If these objects are to be shared among multiple cubes, the objects and cubes must be within the same database.

Databases are immediately subordinate to the Analysis server in the object hierarchy. Thus, after an Analysis server is installed, databases are the first objects to be created. Databases are created in the **Database** dialog box. For more information, see Creating Databases.

After databases are created, data sources are usually the next objects that are created. For more information, see Data Sources.

Databases can be archived and restored by using either Analysis Manager or the **msmdarch** command. For more information, see Archiving and Restoring Databases.

In Analysis Manager, a database is represented by the following icon:

If you are programming with Decision Support Objects (DSO), the class type associated with the database is **clsDatabase**.

In some cases, particularly when you are programming with PivotTable® Service, the database is referred to as the catalog. A noteworthy example is the connection string to an Analysis server in which a database name is specified in the **Initial Catalog** property.

See Also

clsDatabase

Database Security

Analysis Services

Data Sources

A data source contains the information necessary to access source data for an object such as a cube. Although the term *data source* is sometimes used to refer to the source data itself, in this topic it refers to the data source object, which is used by Microsoft® SQL Server™ 2000 Analysis Services to establish connections to the source data.

A data source specifies an OLE DB provider and settings for the other properties in the connection string used to access the source data. The property set varies according to the selected provider. Typically, many of the properties are optional, so the creation of a data source can be accomplished quickly.

Data sources are created in the **Data Link Properties** dialog box. For more information, see Specifying Data Sources.

Analysis Services supports many data sources, including SQL Server 2000 databases and databases created by other products. For more information, including a list of the database products whose databases can be selected when you create a data source, see Specifications and Limits.

When defining a new data source in the **Data Link Properties** dialog box, you can select a database that is accessed through an ODBC driver as the source of data if you specify an OLE DB provider that supports ODBC drivers. One such provider is Microsoft OLE DB Provider for ODBC Drivers, which is supplied with Analysis Services. If you use this provider, you must create a system data source name (DSN) before you create the data source. To create a system DSN, use the ODBC Data Source Administrator in Control Panel.

Data sources are immediately subordinate to the following objects in the object hierarchy:

- Database

  A database can contain multiple data sources, which are shared among the cubes, partitions, and dimensions in the database.

- Cube

  A cube can have only one data source. When a cube is created, its data source is selected from the database's data sources, or a new data source can be created. The cube's partitions can have different data sources than the cube's data source.

    A data source for a linked cube must specify Microsoft OLE DB Provider for Analysis Services or another provider that is compliant with the OLAP section of the OLE DB specification dated March 1999 (2.6).

- Partition

  A partition can have only one data source. It is selected from the database's data sources or created when the partition is created. Each partition in a cube can have a different data source; however, all the data sources must reference sources of data that contain:

  - A fact table with the same structure and columns.

  - The same set of dimension tables that is used in the cube's schema. Among the sources of data, the dimension tables must have the same structure and columns.

In addition, each dimension has a data source. A dimension can be included in a cube only if they have the same data source.

After data sources are created, shared dimensions are typically the next objects that are created. For more information, see Dimensions.

Data sources are also used in incremental updates of cubes and partitions. Before you perform an incremental update, you must create a data source for the source data that will be incorporated into the cube or partition by the incremental update. For more information about incremental updates, see Updating and Refreshing Cube Data.

In Analysis Manager, a data source is identified by the following icon:

If you are programming with Decision Support Objects (DSO), the class type associated with the data source is **clsDataSource**. For more information, see clsDataSource.

See Also

Operational Considerations

Processing Tab (Properties Dialog Box)

Specifying Data Sources

Analysis Services

Dimensions

A dimension is an organized hierarchy of categories, known as levels, that describes data in data warehouse fact tables. Dimensions typically describe a similar set of members upon which the user wants to base an analysis, and they are a fundamental component of cubes. The following topics provide a basic conceptual overview of dimensions.

Topic	Description
Introduction to Dimensions	Provides an overview of the basic concepts of dimensions
Dimension Structure	Describes structural elements of dimensions
Dimension Storage Modes	Describes the differences in how dimensions are stored and settings to determine the dimension storage mode
Dimension Processing	Provides information about the methods available for processing dimensions
Dimension Hierarchies	Provides information about the different ways dimension members can be positioned relative to each other
Ragged Dimension Support	Contains information about how Microsoft® SQL Server™ 2000 Analysis Services can handle missing members in relation to dimension hierarchies
Dimension Characteristics	Describes different dimension characteristics and the dimension varieties that support them
Shared and Private Dimensions	Describes the basic differences between shared and private dimensions and their uses
Changing Dimensions	Describes changing dimensions, their uses, and requirements
Dependent Dimensions	Describes the creation of dependent

	dimensions and identifies their advantages and restrictions
Write-Enabled Dimensions	Describes the creation of write-enabled dimensions and identifies their advantages and restrictions
Balanced and Unbalanced Hierarchies	Describes and depicts differences between balanced and unbalanced hierarchies
Ragged Hierarchies	Details ragged hierarchies, their uses, and supported dimension varieties
Dimension Varieties	Describes and depicts differences between dimension varieties
Regular Dimensions	Provides information about regular dimensions and their variations
Virtual Dimensions	Describes the creation of virtual dimensions and their advantages and restrictions
Virtual Dimensions Created in Version 7.0	Identifies compatibility issues between virtual dimensions created in SQL Server version 7.0 OLAP Services and SQL Server 2000 Analysis Services
Parent-Child Dimensions	Describes the creation of parent-child dimensions and identifies their advantages and restrictions
Data Mining Dimensions	Describes the creation of data mining dimensions and identifies advantages and restrictions to their use

Introduction to Dimensions

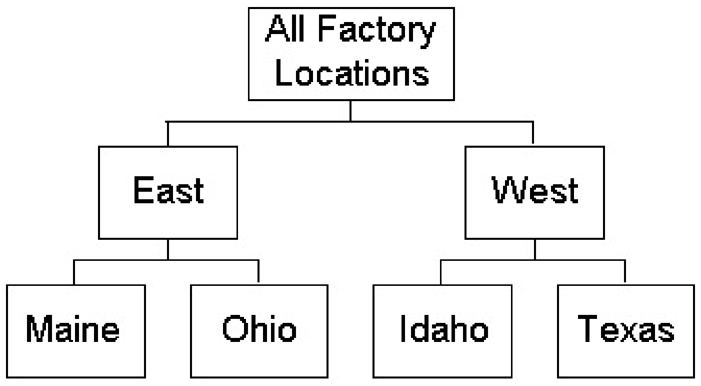
Dimensions are a structural attribute of cubes. They are organized hierarchies of categories and (levels) that describe data in the fact table. These categories and levels describe similar sets of members upon which the user wants to base an analysis.

Dimensions can also be based on OLAP data mining models. They can be used to store the results of a mining model analysis and can be browsed within the context of a virtual cube.

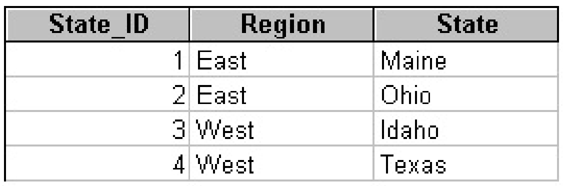
All dimensions are based directly or indirectly on tables, even dimensions based on OLAP mining models. When you create a dimension from a table, you select the columns that define it. The order in which the columns are selected is significant because it affects the placement of members within the dimension's hierarchy. However, the order of members of a dimension created through a mining model analysis is determined by the analysis. This is because the dimension represents the content of the model.

Dimensions are hierarchical, and in most cases their members are arranged in a pyramid-like configuration. The horizontal placement results from column values with the same level in the hierarchy of the dimension, and the vertical placement results from column values having different levels in the hierarchy of the dimension.

For example, the Factory Location dimension has the following members.



This dimension is defined by selecting the **Region** column and then the **State** column from the following table.



State_ID	Region	State
1	East	Maine
2	East	Ohio
3	West	Idaho
4	West	Texas

Note The All Factory Locations member is generated by Microsoft® SQL Server™ 2000 Analysis Services. Generation of an All member is optional. For more information, see (All) Level and All Member.

The primary key of each dimension table joins to a foreign key in a cube's fact table or another dimension table. (An exception is when a virtual cube's dimension uses custom rollup formulas or custom member formulas to determine the values of all of its members.) In the preceding example, the **State_ID** column joins to a foreign key column in the cube's fact table. Key columns are not required in the dimension definition. For more information about the relationships between dimension tables and fact tables, see Cube Structure.

Dimensions categorize the numeric data (that is, measures) in a cube for analysis. For example, if a cube's measure is Production Count, and its dimensions are Product, Time, and Factory Location, end users who access the cube can separate Production Count into various categories of Product, Time, and Factory Location.

The smaller alphanumeric values around the cube are the members of the dimensions. The numeric values within the cube represent the measure, Production Count. These values exist for all cells in the cube but are shown only for those in the foreground.

The total, aggregated value of this cube is the sum of Production Counts for all Products, all Time, and all Factory Locations. This value is the sum of the Production Counts in all 64 of the cube's cells.

The Dimension Wizard enables you to create dimensions quickly and easily. You can use it by itself to create shared dimensions, or, while you are creating a cube with the Cube Wizard, you can invoke the Dimension Wizard to create private or shared dimensions. After a shared dimension is created, you can maintain it in Dimension Editor. After a private dimension is created, you can maintain it in Cube Editor. However, dimensions created by mining model analysis cannot be edited.

After a dimension is created, you must process the dimension. After a dimension is changed or its table is updated, usually you must process the dimension. However, dimension processing can disrupt end users' access to the cubes that include the dimension.

Queries Against Dimension Data

End users can issue queries to analyze either the whole cube or selected portions of it at varying levels of detail. This is accomplished by specifying criteria for each dimension. You can perform these types of operations in Cube Browser. For more information, see Viewing Cube Data.

For example, if the end user chooses to limit the query to Production Counts for hammers, only the front fourth of the cube is retrieved. The Production Counts for pliers, saws, and drills are ignored. This type of operation is called slicing.

The other type of operation that end users perform with dimensions is the combination of drilldown and drillup. This operation determines the level of detail to which the retrieved measure values are separated. For example, if the end user does not drill down at all, the following dataset is returned.

	All Time
All Factory Locations	49005

Notice that no part of the Product dimension appears in the dataset's column headings or row headings. This is because the end user sliced on one of the members of the Product dimension, hammer. A dimension can be used either to slice or to drill down or up, not both.

If the end user drills down into All Factory Locations, a new query is issued, and the following dataset is returned.

		All Time
All Factory Locations		49005
	East	16915
	West	32090

If the end user drills down into the East member, a new query is issued, and the following dataset is returned.

			All Time
All Factory Locations			49005
	East		16915
		Maine	8883
		Ohio	8032
	West		32090

If the end user drills down into All Time, a new query is issued, and the following dataset is returned.

			All Time	1st half	2nd half
All Factory Locations			49005	24981	24024
	East		16915	8962	7953
		Maine	8883	4814	4069
		Ohio	8032	4148	3884
	West		32090	16019	16071

If the end user drills up the East member, a new query is issued, and the following dataset is returned.

		All Time	1st half	2nd half
All Factory Locations		49005	24981	24024
	East	16915	8962	7953
	West	32090	16019	16071

The end user can continue to issue queries by slicing on members in some dimensions and drilling down and up on members in other dimensions.

Object Hierarchy

In the object hierarchy, dimensions are immediately subordinate to the following objects:

- Database

  These dimensions are shared among the cubes in the database.

- Cube

  These dimensions are included in the cube. They may be derived from the shared dimensions in the database or they may be private (that is, unshared).

In the object hierarchy, levels are immediately subordinate to dimensions. The levels of a dimension are created when the dimension is created. Levels are based on the columns in the dimension's definition.

If you are programming with Decision Support Objects (DSO), the class types associated with the dimension are:

- **clsDatabaseDimension**
- **clsCubeDimension**
- **clsPartitionDimension**
- **clsAggregationDimension**

See Also

clsAggregationDimension

clsCubeDimension

clsDatabaseDimension

clsPartitionDimension

Data Mining Models

Dimension Security

Levels

Mining Model Wizard

Dimension Structure

The structure of the dimension you create determines the variety of the dimension. You create dimensions based on dimension table columns, member properties, or from the structure of OLAP data mining models. When you define a dimension, there are a number of possible approaches. Each approach produces a different dimension variety. You can:

- Select one or more columns from a dimension table or joined dimension tables. If you select multiple columns, all columns should be interrelated such that their values can be organized into a hierarchy. To define the hierarchy, sort the columns from most general to most specific. For example, a Time dimension is created from the columns Year, Quarter, Month, and Day. This approach produces a regular dimension (that is, a dimension that is neither a parent-child dimension nor a virtual dimension).

- Select two columns from a single dimension table. One column identifies each component of the dimension's hierarchy, and the other identifies each component's parent. For each row in the table, these two columns identify a parent-child linkage. All the linkages are combined to determine the dimension's hierarchy. For example, a Genealogy dimension is created from the columns Person and Parent. This approach produces a parent-child dimension.

- Select one or more member properties in another dimension. Each member property is based on a column in the other dimension's table. This column contains values that are attributes of another column's values. For example, the Store Type dimension (included in Microsoft® SQL Server™ 2000 Analysis Services) is created from the Store Type member property, which is in the Store dimension and is an attribute of the **store_id** column. This approach produces a virtual dimension.

  An alternative approach to defining a virtual dimension is to directly select columns in another dimension's table. With this approach, member properties are not required.

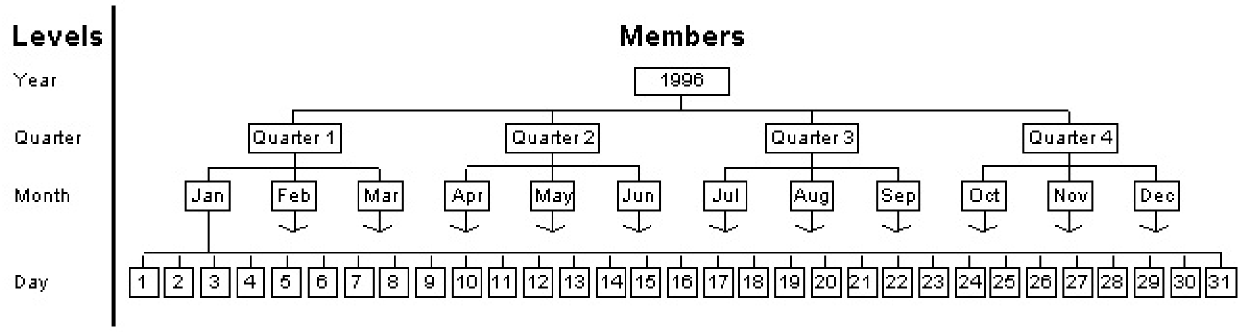
- Choose to create a dimension when creating an OLAP mining model in the Mining Model Wizard. You can create a dimension from the results of the mining model analysis and can also create a virtual cube that contains the dimension and the mining model's source cube.

The columns or member properties in a dimension definition contribute levels to the dimension. Levels are usually ordered by specificity and organized in a hierarchy that allows logical avenues for drilldown. For example, a Time dimension can enable end users to drill down from Year to Quarter, Quarter to Month, and Month to Day. Each drilldown provides greater specificity.

The relationship between the number of columns or member properties in a dimension's definition and the number of levels in the dimension depends on the variety of the dimension. In a regular dimension, each column in the definition of the dimension contributes a level. In a parent-child dimension, the two columns in the definition of the dimension contribute a number of levels that depends on the data in the columns. In a virtual dimension, each member property or column in the definition of the dimension contributes a level.

Each level contains members. Members are the values in the columns or member properties that define the levels. For example, the Quarter level might contain four members: Quarter 1, Quarter 2, Quarter 3, and Quarter 4. However, if data in the table spans more than one year, the Quarter level contains more than four members. For example, if the Year level contains three different members, 1996, 1997, and 1998, the Quarter level contains twelve members.

The relationship between the levels and members in a Time dimension for a single year is shown in the following illustration. (Arrows pointing down indicate members that are not shown. Day members exist for each month, but are shown only for January due to space limitations.)



Using Tabular Browsers

In tabular browsers, members provide the column headings, row headings, and subheadings by which measures are separated and displayed to end users. (In graphical browsers, they provide other types of descriptive labels but serve the same function as in tabular browsers.) For example, in a Time dimension for three years, measures are separated under three headings: 1996, 1997, and 1998. If the end user drills down beneath the Year level, the members of the Quarter level are displayed as subheadings, and the measures are separated further by quarter. If the end user drills down beneath the Quarter level, the members of the Month level are displayed as subheadings beneath the Quarter level headings, and the measures are separated further by month.

See Also

Creating and Maintaining Private Dimensions

Creating Shared Dimensions

Data Mining Models

Dimension Varieties

Levels and Members

Mining Model Wizard

Dimension Storage Modes

A dimension can have one of two storage modes: multidimensional OLAP (MOLAP) or relational OLAP (ROLAP). MOLAP is the default storage mode of a dimension.

The storage mode determines the location and form of a dimension's data. A MOLAP dimension's data is stored in a multidimensional structure on the OLAP server. This structure is created when the dimension is processed. A ROLAP dimension's data is the dimension's table or tables.

MOLAP dimensions provide better query performance than ROLAP dimensions. However, huge dimensions, which are dimensions that have 10 million members or more, cannot support a MOLAP storage mode. If such a dimension's storage mode is MOLAP, processing it produces an error. It is recommended that only huge dimensions have a storage mode of ROLAP.

Note You can create ROLAP dimensions only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition. Very large dimensions, which are generally dimensions that have 5 to 10 million members, can have a MOLAP storage mode.

A dimension's storage mode is set in the **Storage Mode** property in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

Before you set a dimension's **Storage Mode** property to ROLAP, ensure that it meets these requirements:

- The lowest level's **Member Keys Unique** property is set to **True**.
- The dimension does not contain member groups.
- If the dimension is a private dimension, its **Aggregation Usage** property is set to **Standard** if currently set to **Custom**.

- If the dimension is a shared dimension, in all cubes that include the dimension, its **Aggregation Usage** property is set to one of the following values:
    - **Standard**.
    - **Top Level Only**. This value is valid only if the dimension's **All Level** property is set to **Yes**.
    - **Bottom Level Only**.
    - **Top and Bottom Levels**. This value is valid only if the dimension's **All Level** property is set to **Yes**.

IMPORTANT If a dimension's **Storage Mode** is ROLAP, any changes to its source table must be followed by immediate processing of the dimension. Failure to do so may result in inconsistent results to queries of the cubes that include the dimension. To ensure correct processing, include the update of the source table and the processing of the dimension in the same transaction. If you have installed SQL Server 7.0 or later, you can use Data Transformation Services (DTS) to perform the table update and dimension processing as successive tasks connected by a success precedence constraint. For more information, see Processing Objects Using Data Transformation Services. If the dimension is shared, process it with the **Incremental update** option. If the dimension is private, process its cube with the **Refresh data** option.

Virtual and parent-child dimensions always have a MOLAP storage mode.

All regular dimensions that use ROLAP for storage are also changing dimensions. That is, their **Changing** property is set to **True**. When you set the **Storage Mode** property for a regular dimension to ROLAP, its **Changing** property is automatically set to **True**. This value cannot be changed if the **Storage Mode** is ROLAP. For more information, see Changing Dimensions.

ROLAP dimensions do not support slicing in partitions.

Dimension Processing

Two methods are available for processing dimensions: rebuilding the structure of a dimension, and incrementally updating a dimension. These two choices are offered in the **Process a Dimension** dialog box, which appears when you right-click a shared dimension in the Analysis Manager tree pane and then click **Process**.

Rebuilding the Structure of a Dimension

The **Rebuild the dimension structure** option re-creates and loads the dimension. This processing option is required after:

- The structure of the dimension is changed. For example, after you add or remove a level in the hierarchy.
- Relationships between members in the dimension hierarchy are changed. For example, after you redefine sales regions so that cities are now in different regions.

CAUTION If a shared dimension's structure is edited and saved but not processed, it will be processed automatically when any cube that includes the dimension is processed. At that time, any other cubes that include the dimension immediately become unavailable to users and must be processed before they can be used again.

Depending on the type of structure or relationship change, changing dimensions may not require processing with this option. For more information about changing dimensions, see Changing Dimensions.

IMPORTANT When a shared dimension is processed with the **Rebuild the dimension structure** option, all cubes that include the shared dimension immediately become unavailable to users and must be processed before they can be used again.

Incrementally Updating a Dimension

The **Incremental update** processing option updates a dimension when changes have been made to the underlying tables of a dimension, but no structural changes have been made to the dimension itself. Such nonstructural changes can include adding new members, changing existing members, deleting existing members, or changing member properties or custom member options of existing members. For example, you add new customer records to the customer dimension table. A cube that includes a shared dimension remains available to users while the dimension is incrementally updated, and the added dimension members are available in the cube after the update is complete. Because almost any property of a member can be changed by changing the appropriate field in the underlying dimension table, it is possible to have changed member names after the dimension is incrementally updated.

See Also

Updating and Rebuilding Dimensions

Dimension Hierarchies

A hierarchy is the set of members in a dimension and their positions relative to one another. For a dimension created from a data mining model, the hierarchy represents the node structure of the mining model.

Hierarchies are sometimes represented as pyramidal structures. The only exceptions are hierarchies in which all members are at the same level. An example is the hierarchy of the Measures dimension.

From the top of a pyramidal hierarchy to the bottom, the members are progressively more detailed. For example, in a Geography dimension defined with the levels Continent, Country, and City, in that order, the member Europe appears in the top level of the hierarchy, the member France appears in the middle level, and the member Paris appears in the bottom level. France is more specific than Europe, and Paris is more specific than France.

The lower the level of a pyramidal hierarchy, the more members it usually contains. In the preceding example, there are more countries than continents and more cities than countries.

Microsoft® SQL Server™ 2000 Analysis Services supports several types of hierarchies: balanced, unbalanced, and ragged.

Balanced and Unbalanced Hierarchies

In a balanced hierarchy, all branches of the hierarchy descend to the same level, and each member's logical parent is the level immediately above the member. In an unbalanced hierarchy, branches of the hierarchy descend to different levels.

For more information about balanced and unbalanced hierarchies, see Balanced and Unbalanced Hierarchies.

Ragged Hierarchies

In a ragged hierarchy, at least one member's logical parent is not in the level immediately above the member. This can cause branches of the hierarchy to descend to different levels.

For more information about ragged hierarchies, see Ragged Hierarchies.

Dimensions with Multiple Hierarchies

Analysis Services supports dimensions with multiple hierarchies. These dimensions provide similar yet alternate views of cube data. For example, a Time dimension with two hierarchies can have a regular calendar hierarchy and a fiscal calendar hierarchy. In Analysis Services, a dimension with multiple hierarchies is defined as two or more dimensions with names that share the same prefix followed by a period but have different suffixes (for example, dimensions with names of Time.Calendar and Time.Fiscal). The suffix should not equal any current or future level name or member name in the dimension because queries using the dimension may be ambiguous.

See Also

Creating Dimensions with Multiple Hierarchies

Data Mining Models

Dependent Dimensions

Parent-Child Dimensions

Ragged Hierarchies

Ragged Dimension Support

Regular Dimensions

Virtual Dimensions

Ragged Dimension Support

A ragged dimension is a dimension with at least one member whose logical parent is not in the level immediately above the member. For example, a Geography dimension consists of the levels Country, Province, and City. The logical parent of Geneva is Switzerland because Switzerland is not divided into provinces. Because of the missing information for the Province level, the Geography dimension becomes a ragged dimension.

In a ragged dimension's table, the logically missing members, such as the province containing Geneva in the preceding example, can be represented in different ways. The table cells can contain nulls or empty strings, or they can contain the same value as their parent to serve as a placeholder. For example, in the column for the Province level, in rows that contain cities in Switzerland, the value is Switzerland. The nonexistent province of Switzerland is stored as a placeholder because its parent at the Country level is Switzerland.

The representation of placeholders is determined by the placeholder status of child members and the **MDXCompatibilityValue** registry setting (or **MDX Compatibility** connection string property) for PivotTable® Service. If a placeholder has child members that contain data, then its visibility to client applications is dependent on the **MDXCompatibilityValue** registry setting. If a placeholder has no child members, or if all child members of a placeholder are also placeholders with no child members, the placeholder is always skipped, regardless of the **MDXCompatibilityValue** registry setting.

See Also

Dimension Hierarchies

Ragged Hierarchies

MDX Compatibility Property

Using the MDX Compatibility Property

Dimension Characteristics

In addition to supporting several varieties of dimensions, other characteristics can be applied to provide increased functionality, such as ragged hierarchies, to these basic dimension varieties.

The following table displays which dimension characteristics (shown as columns) are supported by the dimension varieties (shown as rows).

	Shared	Private	Changing	Dependent	Write-enabled	Balanced hierarchies	Unbalance hierarchies
Regular	Yes	Yes	Yes*	Yes	No	Yes	No
Virtual	Yes	Yes	Yes**	Yes	No	Yes	No
Parent-Child	Yes	Yes	Yes**	Yes	Yes	No	Yes
Data Mining	Yes	No	Yes**	No	No	Yes	No

*Required for regular dimensions using relational OLAP (ROLAP).
**Required.

Shared and Private Dimensions

A dimension can be created for use in an individual cube or multiple cubes. A private dimension is a dimension created for an individual cube. In Analysis Manager, private dimensions are found in the Cube Editor tree pane. A shared dimension is a dimension that can be used by multiple cubes. In the Analysis Manager tree pane, shared dimensions appear in the Shared Dimensions folder under the database in which they are created. They also appear in the Cube Editor tree pane after they are included in the edited cube.

In the tree panes, a shared dimension is identified by the following icon.

A private dimension is identified by the following icon.

The only exceptions to this rule are virtual dimensions. A shared or private virtual dimension is always identified by the virtual dimension icon. For more information about virtual dimensions, see Virtual Dimensions.

Shared dimensions that share the same data source can be included in any cube or virtual cube in the database. By creating shared dimensions and using them in multiple cubes, you avoid the time-consuming alternative of creating duplicate private dimensions within each of the cubes.

Shared dimensions also enable the standardization of business metrics among cubes. For example, standardized shared dimensions for time and geographic location ensure that data analyzed from different cubes will be similarly organized. This becomes very important when integrating data from different aspects of a business for analysis.

See Also

Creating and Maintaining Private Dimensions

Creating Shared Dimensions

Changing Dimensions

A changing dimension is a dimension that is optimized for frequent changes. A changing dimension permits more types of changes without the subsequent necessity of fully processing the dimension or the cubes that contain it. Full dimension processing interrupts access by end users to any cube including the dimension. Full cube processing also interrupts end users' access to cubes if the cube processing includes full dimension processing. Thus, a changing dimension has the potential advantage of less frequent interruptions of end users' access.

Making a regular dimension into a changing dimension can be beneficial if its table is updated frequently, at unpredictable times, or while end users are connected to cubes that include the dimension. However, making the dimension a changing dimension is advisable only if end users must see these updates soon after they are made. If there is no pressing need to incorporate the dimension table updates in cubes, it is generally better to leave the dimension a non-changing dimension and to process it and the cubes that use it while end users are not connected to the Analysis server.

The following varieties of dimensions are always changing dimensions:

- Virtual
- Parent-child
- Regular dimensions using relational OLAP (ROLAP) storage.

Although changing dimensions can provide greater accessibility than non-changing dimensions, queries that use changing dimensions are somewhat slower.

In a changing dimension, levels below the top level and above the bottom level can be added, moved, renamed, and deleted, and there is no subsequent processing requirement. (The relevant dimension and cube data is updated automatically when the dimension is saved.) An exception is the addition of a level that contains member groups. For more information, see Creating Member

Groups.

Processing Requirements

The processing requirements for member updates vary depending on whether the changing dimension is shared or private.

- Shared

  In a changing shared dimension, members below the top level and above the bottom level can be added, moved, changed, and deleted. (If the dimension has an (All) level, it is the top level. In a parent-child dimension, a leaf member is considered a member of the bottom level for the purpose of determining this processing requirement.) Except for deleted members, the only subsequent processing requirement is to process the dimension with the **Incremental update** option. If members are deleted, the cube must be reprocessed. When the incremental update is complete, cubes that include the dimension are refreshed automatically. Neither the incremental update of the dimension nor the automatic refresh of the cubes interrupts end users' access. When the refresh is complete, end users see the new versions of the cubes.

- Private

  In a changing private dimension, members below the top level and above the bottom level can be added, moved, changed, and deleted; the only subsequent processing requirement is to process the cube that contains the dimension with the **Refresh data** option. (If the dimension has an (All) level, it is the top level. In a parent-child dimension, a leaf member is considered a member of the bottom level for the purpose of determining this processing requirement.) The refresh of the cube does not interrupt end users' access. When the refresh is complete, end users see the new version of the cube.

Write-enabled dimensions whose members are updated in Analysis Manager or through client applications are exceptions; these dimensions and the cubes that contain them do not require subsequent processing.

Creation Requirements

Before you make a changing dimension, unless the dimension is a virtual dimension or a parent-child dimension, be sure that it meets the following requirements:

- The **Member Keys Unique** property of the lowest level must be set to **True**.
- If the dimension is a private dimension, its **Aggregation Usage** property is set to **Standard** if currently set to **Custom**.
- If the dimension is a shared dimension, in all cubes that include the dimension, its **Aggregation Usage** property must be set to one of the following values:
    - **Standard**.
    - **Top Level Only**. This value is valid only if the dimension's **All Level** property is set to **Yes**.
    - **Bottom Level Only**.
    - **Top and Bottom Levels**. This value is valid only if the dimension's **All Level** property is set to **Yes**.

To make a dimension a changing dimension, set its **Changing** property to **True** in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

Dependent Dimensions

A dependent dimension is a dimension that has members that are determined by the members of another dimension. Regular, virtual, and parent-child dimensions can support this dimension characteristic.

Making one dimension dependent on another is advantageous when the cross product of the members of the two dimensions results in a significant percentage of combinations that cannot coexist. For example, a Customer Gender dimension is dependent on a Customers dimension. Fifty percent of the combinations that result from the cross product of the dimensions' lowest-level members cannot coexist because a customer can have only one gender.

To make a dimension dependent on a second dimension, edit the first dimension in Dimension Editor (if the first dimension is shared) or Cube Editor (if the first dimension is private) and select the second dimension in the **Depends on Dimension** property in the properties pane. It is not necessary to specify dependency in both dimensions. For example, if in Dimension A you specify that **Depends on Dimension** is Dimension B, it is not necessary to also specify in Dimension B that **Depends on Dimension** is Dimension A.

All virtual dimensions are dependent dimensions. However, because aggregations do not apply to virtual dimensions, the **Depends on Dimension** property has a different use in virtual dimensions than it does in regular and parent-child dimensions. A virtual dimension's **Depends on Dimension** property identifies the dimension that contains the member properties or columns on which the virtual dimension is based.

In a Microsoft® SQL Server™ 2000 Analysis Services database, multiple dimensions can have the same value for their **Depends on Dimension** property.

Dependent Dimensions and Aggregation Design

For certain dimension varieties, if a dimension is set as a dependent dimension, its aggregation design characteristics may be affected. If a regular or parent-child dimension is a dependent dimension, the design of aggregations for partitions that include the dimension is optimized according to the dimension on which the regular or parent-child dimension is dependent.

When aggregations are designed for partitions that include dependent dimensions, aggregations for the nonexistent member combinations are excluded from the estimated storage size. For example, if Pat Coleman is a male, the estimated storage size excludes aggregations for the intersections of the Pat Coleman member and the Female member. Because aggregations for these intersections are not created when the partition is processed (regardless of whether the dimensions are dependent), the actual storage size is closer to the estimated storage size if the dimensions are dependent.

Note If you stop the aggregation design process before the aggregations for the nonexistent member combinations are reached, the reduction in the estimated storage size does not occur.

In Analysis Manager, aggregation design is performed in the **Set aggregation options** step in the Storage Design Wizard and the Usage-Based Optimization Wizard.

See Also

Dimension Hierarchies

Virtual Dimensions

Write-Enabled Dimensions

A write-enabled dimension allows end users to modify the contents of the dimension and see the immediate impact on the cube. This ability provides added analytical options to end users. For example, in an Employee dimension, an end user can move the employee members beneath different managers to assess the changes on measures such as Budget and Staffing Level. Only parent-child dimensions support this dimension characteristic.

Note This feature is available only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

End users can update a write-enabled dimension by using client applications. Administrators can update members of a write-enabled dimension by using Analysis Manager. In a write-enabled dimension, end users and administrators can change, move, add, and delete members. They can also update member property values. In Analysis Services, these updates are referred to collectively as dimension writeback.

Note Unlike updates to a write-enabled cube, which are stored in a writeback table separate from the cube's source tables, updates to a write-enabled dimension are recorded directly in the dimension's table. Also, if the write-enabled dimension is included in a cube with multiple partitions where some or all of their data sources have copies of the dimension table, only the original dimension table is updated during a writeback process.

Write-enabled dimensions and write-enabled cubes have different but complementary features. A write-enabled dimension gives end users the ability to update members, whereas a write-enabled cube gives them the ability to update cube cells. Using these two features in combination is optional. For example, a write-enabled dimension can be included in a cube that is not write-enabled. Different procedures are used to write-enable dimensions and cubes and to maintain their security. The only kind of dimension that can be write-enabled is a parent-child dimension.

The only end users who can update a write-enabled dimension are those in cube roles granted read/write access to the dimension. For each role, you can control which members can and cannot be updated. In order for end users to update write-enabled dimensions, their client application must support this capability.

For shared write-enabled dimensions, an administrator can update the members and associated member property values of a write-enabled dimension in the dimension members pane of Dimension Editor or in Dimension Browser. For private write-enabled dimensions, Dimension Browser can be opened from Cube Editor to update the members and associated member property values of a write-enabled dimension. These actions require the write-enabled dimension to be included in a cube that was processed since the dimension last changed.

Dimension writeback is not supported in distributed partitioned cubes. You cannot write-enable a dimension that is included in a distributed partitioned cube, and you cannot add a write-enabled dimension to a distributed partitioned cube. For more information about distributed partitioned cubes, see Distributed Partitioned Cubes.

To write-enable a dimension, set its **Write-enabled** property to **True** in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

All write-enabled dimensions are also changing dimensions. That is, their **Changing** property is set to **True**. This value cannot be changed. Write-enabled dimensions have the same processing requirements as changing dimensions except that dimension writeback does not require subsequent processing.

See Also

Changing Dimensions

Cube Role Manager

Cube Editor (Schema View)

Dimension Editor (Schema View)

Linked Cubes

Remote Partitions

Write-Enabled Cubes

Balanced and Unbalanced Hierarchies

In a balanced hierarchy, all branches of the hierarchy descend to the same level, and each member's logical parent is the level immediately above the member. In the preceding Geography dimension example, if the country members are limited to France and Germany, the dimension's hierarchy is balanced. All branches of the hierarchy terminate at cities at the City level. The parent of each city is either France or Germany, and Europe is the parent of both France and Germany.

In an unbalanced hierarchy, branches of the hierarchy descend to different levels. For example, an Organization dimension contains a member for each employee in a company. The CEO is the top member in the hierarchy, and the division managers and executive secretary are immediately beneath the CEO. The division managers have subordinate members but the executive secretary does not.

In Analysis Services, balanced hierarchies are supported in regular and virtual dimensions. Unbalanced hierarchies are supported only in parent-child dimensions.

It may be impossible for end users to distinguish between unbalanced and ragged hierarchies. However, you can employ different techniques and properties to support these two types of hierarchies.

See Also

Dimension Hierarchies

Ragged Dimension Support

Ragged Hierarchies

MDX Compatibility Property

Ragged Hierarchies

In Microsoft® SQL Server™ 2000 Analysis Services, ragged hierarchies are supported in regular and parent-child dimensions. A ragged hierarchy is unbalanced if the hierarchy is in a parent-child dimension.

In a ragged hierarchy, the logical parent member of at least one member is not in the level immediately above the member. This can cause branches of the hierarchy to descend to different levels.

For example, in a Geography dimension defined with the levels Continent, Country, and City, in that order, the member Europe appears in the top level of the hierarchy, the member France appears in the middle level, and the member Paris appears in the bottom level. France is more specific than Europe, and Paris is more specific than France. To this regular hierarchy, the following changes are made:

- The Country Switzerland is added to France and Germany.
- Cities in Switzerland are added.
- The level Province is added between Country and City and populated with members except under Switzerland, which has no provinces.

Because of the alterations, the hierarchy of the dimension is now ragged. The parent of the city Geneva is the country Switzerland, which is not in the level immediately above Geneva.

To make a hierarchy ragged, various methods can be selected in the **Hide Member If** property of a level in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

Alternatively, if the ragged dimension is a parent-child dimension, the **Skipped Levels Column** property of the level is used to support the ragged structure. If you use this property, ensure that the dimension table contains a column that stores the number of intervening levels between each member and its parent.

Both properties can cause the logically missing members to be hidden from end users as they browse cubes. Neither property is supported in virtual dimensions.

Ragged hierarchies can appear ragged to end users only if certain criteria are met. That is, only if the criteria are met can members whose logical parents are not in the level immediately above them nevertheless appear immediately beneath their logical parents. These criteria are:

- The property that supports ragged hierarchies is correctly set. In a regular dimension, the level's **Hide Member If** property must be set to a value other than **Never hidden**. The selected value must be appropriate for the contents of the dimension's table. In a parent-child dimension, the level's **Skipped Levels Column** property must be set to a column containing integer values that indicate the number of intervening levels between members and their parents. These properties are set in Dimension Editor if the dimension is shared or Cube Editor if the dimension is private.

- The client application supports the display of ragged hierarchies.

- The **MDX Compatibility** property in the connection string from the client application to the Analysis server is set to 2, or the **MDXCompatibilityValue** registry setting for PivotTable® Service is set to 2.

See Also

Dimension Hierarchies

Ragged Dimension Support

Balanced and Unbalanced Hierarchies

MDX Compatibility Property

Dimension Varieties

Microsoft® SQL Server™ 2000 Analysis Services includes several varieties of dimensions. The most general distinctions are among regular, virtual, parent-child, and data mining dimensions. All dimensions are regular, virtual, parent-child, or data mining; each of these varieties can have other dimension characteristics that are determined by property settings for the dimension.

For more information about the various dimension characteristics available for dimension varieties, see Dimension Characteristics.

Regular Dimensions

A regular dimension is a dimension that is not a virtual dimension, parent-child dimension, or data mining dimension. Unlike virtual dimensions, regular dimensions have associated aggregation data in the cubes in which they are used. Unlike parent-child dimensions, whose hierarchies are unbalanced, the hierarchies in regular dimensions are either balanced or ragged. (All dimensions are regular, virtual, parent-child, or data mining.)

A regular dimension contains a number of levels equal to the number of columns selected during its definition. These levels are usually sorted from most general to least general. Virtual dimensions are similar in this respect, but parent-child dimensions always contain a single meta data level that usually produces multiple levels in the end users' view.

Unlike a parent-child dimension, which is always based on a single table, a regular dimension can be based on multiple, joined tables. If a regular dimension is based on multiple tables, adding the dimension to a cube causes it to adopt a snowflake schema if it does not already have one. (If the cube already has a snowflake schema, it retains a snowflake schema.) If a regular dimension is based on a single table, and the dimension is added to a cube with a star schema, the cube retains a star schema.

Regular dimensions can be either shared or private. For more information about shared and private dimensions, see Shared and Private Dimensions.

You can create regular dimensions by using either the Dimension Wizard or Dimension Editor for shared regular dimensions, or Cube Editor to create private regular dimensions. For more information about the Dimension Wizard and Dimension Editor, see Dimension Wizard and Dimension Editor - Schema View.

To create a shared dimension using the Dimension Wizard

Virtual Dimensions

A virtual dimension is a logical dimension based on the columns from a physical dimension. These contents can be either:

- Member properties in the physical dimension. Member properties must be created before the virtual dimension. For more information, see Member Properties.

  -or-

- Columns in the tables of the physical dimension.

For example, the Store Name level of the Store dimension has a member property Store Sqft that identifies the area of the store in square feet. Using the Store Sqft member property, you can create a virtual dimension named Store Size in SqFt. This virtual dimension can be added to any cube that also contains the Store dimension.

Note You can add a virtual dimension to a cube only if the dimension that supplies the member properties or columns on which the virtual dimension is based is also included in the cube.

End users can use a virtual dimension like any other dimension. If a virtual dimension is based on member properties, the virtual dimension can enable end users to analyze cube data based on the member properties. Furthermore, the depth of the virtual dimension in terms of the number of levels it has depends on the number of member properties that are used to define it.

Adding a virtual dimension to a cube does not increase the cube's size because a virtual dimension, unlike a regular or parent-child dimension, does not have aggregation data. Virtual dimensions do not affect cube processing time because they are calculated in memory when needed. However, queries that use virtual dimensions can be slower than queries that use regular or parent-child dimensions.

Microsoft® SQL Server™ 2000 Analysis Services provides the Dimension Wizard for easy creation of virtual dimensions based on member properties.

To create a virtual dimension based on member properties

Virtual Dimensions Created in Version 7.0

Virtual dimensions created in Microsoft® SQL Server™ 2000 Analysis Services are significantly different from virtual dimensions created in SQL Server version 7.0 OLAP Services. Other topics in this documentation describe virtual dimensions created in SQL Server 2000 Analysis Services, unless otherwise noted. This topic describes how virtual dimensions created in SQL Server 2000 Analysis Services are different from those created in version 7.0.

In SQL Server 2000 Analysis Services, you can continue to use virtual dimensions created in version 7.0. However, some limitations apply to version 7.0 virtual dimensions. (These limitations apply to all virtual dimensions in version 7.0.) Also, virtual dimensions created in SQL Server 2000 Analysis Services generally provide better query performance. If you want to overcome these limitations or to gain the performance improvement of SQL Server 2000 Analysis Services, re-create your version 7.0 virtual dimensions in SQL Server 2000 Analysis Services. For more information, see Creating Virtual Dimensions.

The following table summarizes the limitations of virtual dimensions created in version 7.0.

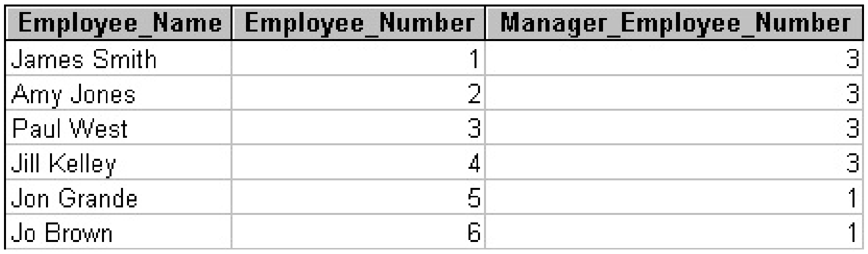
A virtual dimension	Created in SQL Server 2000 Analysis Services	Created in version 7.0
Has a maximum limit of 759 members	No	Yes
Can have multiple levels (excluding the (All) level)	Yes	No
Must be based on member properties	No	Yes
Can be edited in Dimension Editor	Yes	No
Can be defined as private using Analysis Manager	Yes	No

Can be changed to a regular dimension	Yes	No
Can contain member properties	Yes	No
Supports the following functions in Multidimensional Expressions (MDX): **Cousin**, **ClosingPeriod**, **LastPeriods**, **OpeningPeriod**, **ParallelPeriod**, **PeriodsToDate**, **WTD**, **MTD**, **QTD**, **YTD**	Yes	No
Can be used to define an OLAP data mining model	Yes	No

Parent-Child Dimensions

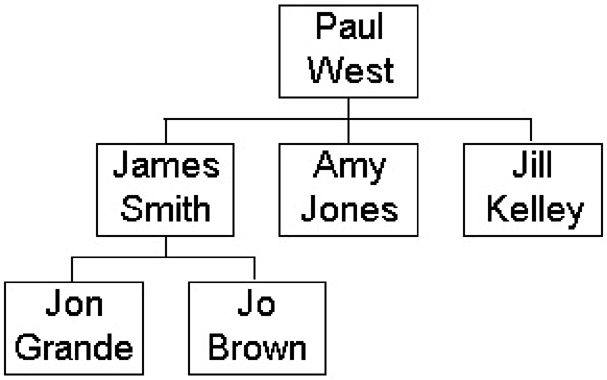
A parent-child dimension is based on two dimension table columns that together define the lineage relationships among the members of the dimension. One column, called the member key column, identifies each member; the other column, called the parent key column, identifies the parent of each member. This information is used to create parent-child links, which are then combined into a single member hierarchy that represents a single meta data level.

For example, in the following **Employee** table, the column that identifies each member is **Employee_Number**. The column that identifies the parent of each member is **Manager_Employee_Number**. (This column stores the employee number of each employee's manager.)



Employee_Name	Employee_Number	Manager_Employee_Number
James Smith	1	3
Amy Jones	2	3
Paul West	3	3
Jill Kelley	4	3
Jon Grande	5	1
Jo Brown	6	1

These columns can be used to define a parent-child dimension that contains the following member hierarchy. (The hierarchy mirrors an organization chart of the employees in the **Employee** table.)



Both columns must have the same data type. Both columns must be in the same table.

Note By default, any member whose parent key equals its own member key, null, 0 (zero), or a value absent from the column for member keys is assumed to be a member of the top level (excluding the (All) level).

By default, a top-level member (ignoring the (All) level) is identified by its parent key, which equals its own member key, null, 0 (zero), or a value not contained in the column for member keys. For example, in the preceding illustration, the only top-level member is Paul West. Both the **Manager_Employee_Number** value and **Employee_Number** value for Paul West are 3. These values are equal because Paul West is specified as the manager of Paul West. By default, Paul West would also be a top-level member if the **Manager_Employee_Number** value for Paul West were null, 0 (zero), or a value not contained in the **Employee_Number** column.

You can set the criteria for identifying top-level members by using the level's **Root Member If** property in the properties pane of Dimension Editor (if the dimension is shared), or Cube Editor (if the dimension is private).

When you define a parent-child dimension, you can also select a third column to provide member names, which are displayed to end users as they browse cubes. This third column, the member name column, defaults to the member key column. If you want to display an alternate value, a different column can be chosen. In the preceding illustration, the employee names would be displayed only if the member name column was set to:

```
"Employee"."Employee_Name"
```

The depth of a parent-child dimension can vary among its hierarchy's branches. For example, in the preceding illustration, the James Smith branch has lower-level members, but the Amy Jones and Jill Kelley branches do not. Therefore, the hierarchies of parent-child dimensions are usually unbalanced.

Unlike regular and virtual dimensions, which are defined with a number of levels that determines the number of levels seen by end users, a parent-child dimension is defined with a single level of a special type that usually produces multiple levels seen by end users. The number of displayed levels depends on the contents of the columns that store the member keys and the parent keys. This number can change when the dimension table is updated and the cubes using the dimension are subsequently processed.

You can use the Dimension Wizard to create parent-child dimensions. You cannot use Dimension Editor. You can use Cube Editor to create parent-child dimensions only if you start the Dimension Wizard from within Cube Editor. In the second step of the Dimension Wizard, select **Parent-Child Dimension: Two related columns in a single dimension table**.

After you create a parent-child dimension, you can edit it in Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private). In Dimension Editor or Cube Editor, you can access the **Level Naming Template** dialog box, in which you can specify the level names to be displayed to end users.

A parent-child dimension's table is graphically represented with a join between the column containing the members' keys and the column containing the keys of the members' parents. This join is visible in the Dimension Wizard and the **Schema** tab of Dimension Editor and Cube Editor.

IMPORTANT If a parent-child dimension is included in a cube with a fact table that has rows associated with the dimension's nonleaf members, you must set the dimension's **Members With Data** property to **Nonleaf data visible** or **Nonleaf data hidden**. Otherwise, processing the cube fails.

The **Members With Data** property indicates whether nonleaf members of a parent-child dimension are allowed to have associated fact table data. By default, nonleaf members are not allowed to have associated fact table data, so the property is initially set to **Leaf members only**. The related **Data Member Caption Template** property controls the names of data members when the **Members With Data** property is set to **Nonleaf data visible**. For more information about these properties, see Properties Pane (Cube Editor Data View) and Properties Pane (Dimension Editor Data View).

Parent-child dimensions are the only kind of dimension that you can write-enable. All parent-child dimensions are also changing dimensions. That is, their **Changing** property is set to **True**. You cannot change this value.

The storage mode of a parent-child dimension is always multidimensional OLAP (MOLAP).

See Also

Changing Dimensions

Dimension Wizard

Write-Enabled Dimensions

Data Mining Dimensions

Although data mining dimensions are shared dimensions, they differ from other types of shared dimensions in several ways. Unlike other types of shared dimensions, data mining dimensions cannot be created in Dimension Editor, and they must be based on OLAP data mining models. Furthermore, they cannot be edited after they have been created, they do not support dimension security through database or cube roles, and they can be included only in virtual cubes.

Data mining dimensions can be created in either the Dimension Wizard or the Mining Model Wizard. If you use the Dimension Wizard to create your data mining dimension, you can use an existing OLAP mining model. If you use the Mining Model Wizard to create a data mining dimension, you can create the new dimension at the same time that you create the new OLAP mining model it is based on. When creating a dimension with the Mining Model Wizard, you also have the option of creating a virtual cube to contain the new data mining dimension and the source cube of the mining model. If you choose not to create a virtual cube to contain the data mining dimension while you are in the Mining Model Wizard, you can use the Virtual Cube Wizard later to add it to an existing or new virtual cube.

To view the members of a data mining dimension, use Dimension Browser. Excluding the top node of a mining model that is based on either the Microsoft® Decision Trees or the Microsoft Clustering algorithm, each level member of the dimension represents the rule corresponding to a node in the mining model. You can view the Multidimensional Expressions (MDX) statement used to generate the rule in the custom member formulas pane of Dimension Browser.

IMPORTANT Data mining dimensions can contain custom rollup formulas and custom members; therefore, data mining dimensions cannot be used with regular cubes containing distinct count measures. This limitation does not apply to virtual cubes.

See Also

Introduction to Data Mining Models

Creating OLAP Data Mining Models

Mining Model Wizard

Create a Dimension and Virtual Cube (Mining Model Wizard)

Dimension Browser

Virtual Cubes

Virtual Cube Wizard

Viewing Data Mining Models

Custom Rollup Formulas and Custom Member Formulas

Data Mining Algorithms

Analysis Services

Levels and Members

A level is an element of a dimension hierarchy. Levels describe the hierarchy from the highest (most summarized) level to the lowest (most detailed) level of data.

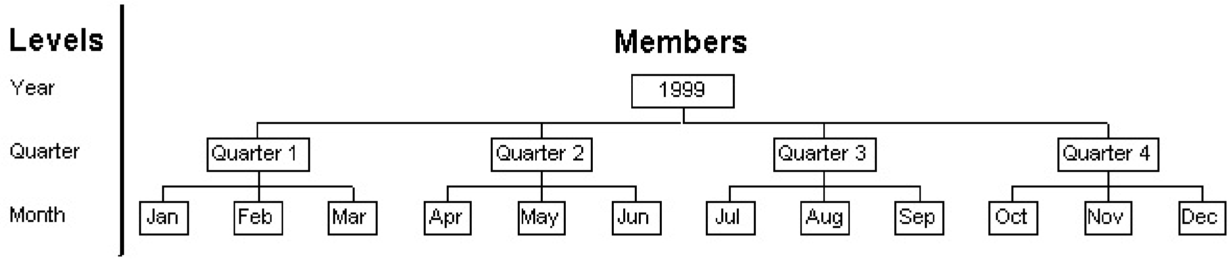
Levels exist only within dimensions. They are based on columns in a dimension table or member properties in a dimension.

Levels are defined within a dimension to specify the contents and structure of the dimension's hierarchy. That is, the level definitions determine the members that are included in the hierarchy and their positions relative to one another within the hierarchy.

Levels are created when you create a dimension in the Dimension Wizard, Dimension Editor, or Cube Editor. After you create a dimension, you can maintain its levels in Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private). In the editors you can set the properties of the levels.

After shared dimensions and their levels are created, measures are typically the next objects to be created. However, they are created in the process of creating the cubes that contain them.

For example, a Calendar dimension contains the levels Year, Quarter, and Month. The relationship between the levels and members of the Calendar dimension (a regular dimension) is shown in the following diagram.



For example, a Calendar dimension contains the levels Year, Quarter, and Month.

Year	Quarter	Month

1999	Quarter 1	Jan
1999	Quarter 1	Feb
1999	Quarter 1	Mar
1999	Quarter 2	Apr
1999	Quarter 2	May
1999	Quarter 2	Jun
1999	Quarter 3	Jul
1999	Quarter 3	Aug
1999	Quarter 3	Sep
1999	Quarter 4	Oct
1999	Quarter 4	Nov
1999	Quarter 4	Dec

You can define a level in one of three ways, depending on the variety of dimension in which the level is defined. For regular dimensions, select a column from the dimension table; this column supplies the members, or components, of the level.

When working with parent-child dimensions, it is important to distinguish between the level object and a level in the hierarchy. A parent-child dimension always contains only one level object, but the hierarchy of the dimension usually contains multiple levels. For a parent-child dimension, select two columns from the dimension table. One column identifies the members of the dimension, and the other column identifies the parents of the members. For each row in the table, the two columns identify a parent-child linkage. All the linkages are combined to determine the hierarchy of the dimension. The column that contains the member identifiers is the column that supplies all of the members of the dimension.

For a virtual dimension, select a member property in another dimension or in a column in the table of another dimension. This member property or column supplies the members of the level.

The identification of members to be included in a level is controlled by its **Member Key Column** property. This is the same column that supplies the level's members as described earlier in this topic. (The initial value of this property is set when you create a dimension with the Dimension Wizard.)

In a regular or virtual dimension, the vertical positions of members within the dimension's hierarchy are controlled by the order of the levels in the dimension's definition. Each level in the definition produces a level in the hierarchy. The levels' vertical order in the definition matches the levels' order in the hierarchy. The horizontal position of a member is determined by the level in which it is included.

In parent-child dimensions, the vertical positions of members are determined differently. In a parent-child dimension, only a single level can be defined (besides the optional (All) level), but it usually produces multiple levels in the hierarchy. Members' vertical positions are determined by the level's **Parent Key Column** and **Root Member If** properties. Members' horizontal positions within a level are determined by the level's **Order By** property.

The (All) level is a special kind of level. Except in virtual dimensions, the (All) level is optional. If defined, it is the highest level in the dimension. It contains a single member whose value is the aggregation of the values of the members in the immediately subordinate level.

Levels are immediately subordinate to the following objects in the object hierarchy:

- Dimension immediately beneath a database (shared dimension)

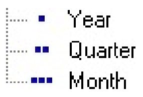
  These levels are in dimensions that are shared among the cubes in the database.

- Dimension immediately beneath a cube (shared or private dimension)

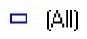
  These levels are in dimensions that are included in the cube. These dimensions may be derived from the shared dimensions in the database or they may be private (that is, unshared).

Member properties are immediately subordinate to levels. Member properties are optional objects that provide end users with additional information about members.

In Analysis Manager, levels are identified by icons containing very small squares. The number of squares indicates the level's position in the dimension's definition. Except for the (All) level, the highest level's icon has one square, the second level's icon has two squares, and so on. For example, the following three icons represent the top three levels in a dimension.



The icon for the (All) level is not displayed in Dimension Editor or Cube Editor but is displayed in some dialog boxes. The icon for the (All) level looks like this.



If you are programming with Decision Support Objects (DSO), the class types associated with the level are **clsAggregationLevel**, **clsDatabaseLevel**,**clsCubeLevel**, and **clsPartitionLevel**.

See Also

(All) Level and All Member

clsAggregationLevel

clsCubeLevel

clsDatabaseLevel

clsPartitionLevel

(All) Level and All Member

The (All) level is a special kind of level. Its name is always (All). If a dimension contains an (All) level, the (All) level is the highest level. It contains only one member whose value is the aggregation of the values of all members in the immediately subordinate level. Because the member in the (All) level is at the top of the dimension's hierarchy, the member's value is the consolidated aggregation of the values of all members in the hierarchy.

By default all dimensions contain an (All) level. However, you can remove the (All) level except from virtual dimensions in which it is required.

The single member of the (All) level is called the All member. The All member is a system-generated member that is not contained in the dimension table. You can change the name of the All member. By default, its name is the word All followed by a space and the name of the dimension.

To add or remove the (All) level from a dimension or to change the name of the All member, use the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private). Inclusion of the (All) level is controlled by the dimension's **All Level** property. The name of the All member is set in the dimension's **All Caption** property.

Whether a dimension contains an (All) level can affect queries on cubes that contain the dimension. When a query neither slices explicitly on a particular dimension nor projects that dimension on an axis, by default the query slices on the dimension's default member. If a dimension contains an (All) level, by default the default member is the All member. Slicing by the All member does not limit the retrieved data, so the query retrieves a dataset that is complete for the dimension in question. However, if the dimension does not contain an (All) level, or if a default member other than the All member is specified, the query retrieves a dataset that is incomplete for the dimension in question. (An exception is when the dimension does not contain an (All) level, but the dimension's top level contains only one member that serves as the default member. This member functions similarly to an All member insofar as it does not limit the retrieved data.) The fact that the dataset is incomplete may not be obvious to the end user who issues the query. For this reason, it is usually a good practice to retain the (All) level within a dimension especially when no default member is selected.

The default member of a dimension is specified in the **Default Member** property in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private). Within a dimension, a different default member can be selected for each role. For more information, see Custom Rules in Dimension Security.

Data Members

In Analysis Services, an assumption is usually made about the content of members. Leaf members contain data derived from underlying data sources;nonleaf members contain data derived from aggregations performed on child members.

In a parent-child dimension, however, some nonleaf members may also have data derived from underlying data sources, in addition to data aggregated from child members. For these nonleaf members in a parent-child dimension, special system-generated child members can be created that contain the underlying fact table data. Referred to as data members, they contain a value directly associated with a nonleaf member that is independent of the summary value calculated from the descendants of the nonleaf member.

Data members are available only to parent-child dimensions, and only if the parent-child dimension allows nonleaf members with data. This feature can be enabled with Dimension Editor by setting the **Members with Data** property to **Nonleaf data visible** or **Nonleaf data hidden**, or with the Decision Support Objects (DSO) library by using the **MembersWithData** property of a **Dimension** object with a **SubClassType** of **sbclsParentChild**. If the parent-child dimension has data in fact tables associated with nonleaf members and the **Members with Data** property is set to **Leaf members only**, a processing error will occur.

Setting the **Members with Data** property in Dimension Editor to **Nonleaf data hidden** allows nonleaf members to have data, but this setting overrides the normal aggregation behavior for nonleaf members. This can cause confusion, because the values for nonleaf members are not derived from the aggregation of child members but from underlying fact table data for the nonleaf member.

Setting the **Members with Data** property in Dimension Editor to **Nonleaf data visible** also allows nonleaf members to have data, but this setting creates an additional system-generated child member that contains the underlying fact table data for the nonleaf member. This setting does not override the normal aggregation behavior for nonleaf members; the data member is treated as a child member for the purposes of aggregation. Although a custom rollup formula can be used to override this behavior, the Multidimensional Expressions (MDX) **DataMember** function gives you the ability to access the value of the associated data member regardless of the aggregation behavior.

The benefit of this functionality is not readily apparent for most client applications. However, in certain specific situations data members are indeed very beneficial. For example, the following diagram shows a dimension, representing gross sales volume of products, with three levels. The first level shows the gross sales volume for all salespersons. The second level contains the gross sales volume for all sales staff by sales manager, and the third level contains the gross sales volume for all sales staff by salesperson.

In the case of the Sales Manager 1 member, aggregating the values of the Salesperson 1 and Salesperson 2 members would ordinarily derive the value of the member. However, because Sales Manager 1 also can sell products, that member may also contain data derived from the fact table because there may be gross sales associated with Sales Manager 1.

The individual commissions for each sales staff member can vary. For sales managers, two different scales are used to compute commissions for their individual gross sales, as opposed to the total of gross sales generated by their salespersons. In this case, the ability to access the underlying fact table data for nonleaf members becomes important. The MDX **DataMember** function can be used to retrieve the individual gross sales volume of the Sales Manager 1 member, while a custom rollup expression can be used to exclude the data member from the aggregated value of the Sales Manager 1 member, providing the gross sales volume of the salespersons associated with that member.

See Also

Parent-Child Dimensions

Dimension Editor - Schema View

Dimension Interface

MembersWithData

Member Names and Member Keys

The member names of a level can be derived from a column different from the keys of the members. Thus, a level can be derived from two columns.

Member names are displayed to end users when they browse cubes that contain the dimension. The column that supplies the member names of a level is selected in the **Member Name Column** property of the level.

In Microsoft® SQL Server™ 2000 Analysis Services, member keys identify members within a level. The column that supplies the member keys of a level is selected in the **Member Key Column** property of the level.

When you create a dimension, the column that you select to define a level is recorded in both the **Member Name Column** property and the **Member Key Column** property. (If you create a virtual dimension based on member properties, the column is selected indirectly. The source column of the member property is used for the value of both properties.) You can later change one of these properties so that they reference different columns. Both of these properties can be set when you create a dimension with Dimension Wizard. Later, the values of these properties can be changed in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

Custom Rollup Operators

Custom rollup operators provide a simple way to control how level member values are rolled up to their parent's values. When custom rollup operators are assigned a column, either when creating them as an optional feature of new parent-child dimensions in Dimension Wizard or adding them to existing dimensions in Dimension Editor or Cube Editor, the contents of that column are used as the custom rollup operator for each member. This custom rollup operator is applied to the member when evaluating the value of the member's parents.

In Dimension Editor, custom rollup operators are enabled by setting the **Unary Operators** property of the level to **True**. Values for the custom rollup operators are stored in the column listed by the **Define Unary Operator Column** dialog box and are applied to each member.

Custom rollup operators provide similar but simplified functionality of custom member formulas. In comparison to custom member formulas, which use Multidimensional Expressions (MDX) expressions to determine how members are rolled up, the custom rollup operator uses simple math operators to determine how the value of a member affects the parent. Like custom member formulas, the value of the custom rollup operator is unique for each level member.

In terms of precedence, a level's custom rollup operators override the custom rollup expression of the previous level. However, the custom member formulas of the preceding level override the custom rollup operators of a level.

Custom rollup operators can be enabled for both shared and private dimensions. To enable custom rollup operators in a level for a dimension, use its **Unary Operators** property in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private). Clicking the edit (**...**) button beside this property's value displays the **Define Unary Operator Column** dialog box in which you select or create a column to store the formulas. After you close this dialog box, if the shared dimension is a write-enabled, parent-child dimension, you can select values for the UNARY_OPERATOR member property in the custom member formula pane of Dimension Editor or Dimension Browser. (To perform this action, make sure that the write-enabled dimension is included in a cube that was processed since the dimension last changed and process the cube after changes have been made.) If the dimension is not write-enabled, you must use a tool other than Analysis Manager to insert the formulas into the column.

Note If a cube has a measure with its **Aggregate Function** property set to **Distinct Count**, adding a custom rollup operator or expression to a level will cause the structure of the cube to become invalid.

See Also

Custom Rollup Formulas and Custom Member Formulas

Properties Pane (Cube Editor Data View)

Using Custom Rollup Operators

Custom Rollup Formulas and Custom Member Formulas

Custom rollup formulas and custom member formulas are expressions, written in Multidimensional Expressions (MDX), that determine the cube cell values associated with members. A custom rollup formula applies to all the members (except calculated members) in a level. A custom member formula applies to a single member.

Custom rollup formulas and custom member formulas are different from calculated members. Custom rollup formulas and custom member formulas apply to members that exist in dimension tables. In contrast, calculated members are not stored in dimension tables. Calculated members provide members in addition to those in the tables.

Custom rollup formulas and custom member formulas use similar kinds of MDX expressions.

Both custom rollup formulas and custom member formulas override the aggregate functions associated with measures. For example, before a custom rollup formula is specified, a measure using the **Sum** aggregate function has the following values for the following members of the Time dimension:

- 1997: 2100
    - Quarter 1: 700
    - Quarter 2: 500
    - Quarter 3: 100
    - Quarter 4: 800
- 1998: 1500
    - Quarter 1: 600

- Quarter 2: 200
- Quarter 3: 300
- Quarter 4: 400

For the Year level, the following custom rollup formula is specified:

```
Time.CurrentMember.LastChild
```

This custom rollup formula overrides the **Sum** aggregate function and produces the following new values for the Year members:

- 1997: 800
- 1998: 400

The values for the Quarter members are unchanged.

Custom member formulas operate in a similar manner, but each affects only a single member. The value of the custom member is supplied by the custom member formula. For example, the following custom member formula can be used to supply the value for the Quarter 4 child member of the 1998 member in the Time dimension.

```
Time.[Quarter 3] * 1.5
```

Custom member formulas are stored in a column of the dimension table. When you enable custom member formulas, a dialog box appears in which you select or create this column. This procedure is summarized later in this topic.

To apply a custom rollup formula to only some members of a level, use the **IIf** and **RollupChildren** functions. The **RollupChildren** function can roll up the children of a specified member, using unary operators specified in the function. For example, to apply the custom rollup formula Sales * 0.10 to only the children of SalesPersons, type the following custom rollup formula:

```
IIf(Employees.CurrentMember.Parent.Name = "SalesPersons", Sales *
    RollupChildren(Employees.CurrentMember,
    Employees.CurrentMember.Properties("UNARY_OPERATOR"))
```

For more information about these functions, see IIf and RollupChildren.

In terms of the order of evaluation, if a level has both a custom rollup formula and custom member formulas, the custom member formulas override the custom rollup formula. Calculated members are resolved before custom rollup formulas and custom member formulas are resolved. If a cube contains multiple dimensions with custom rollup formulas or custom member formulas, the formulas are resolved in the order that the dimensions were added to the cube. You can view and change this order in the Cube Editor tree pane. For more information about the order of evaluation for various formulas, see Understanding Pass Order and Solve Order.

To specify a custom rollup formula in any level except an (All) level, use the **Custom Rollup Formula** property of the level. To specify a custom rollup formula in an (All) level, use the **All Member Formula** property of the dimension. You can access these properties in the properties pane of Cube Editor. Clicking the edit (**...**) button beside the values of these properties displays MDX Builder, in which you can construct the custom rollup formula.

A custom rollup formula can be specified in either a shared or private dimension. If it is specified in both, the custom rollup formula in a cube's private dimension takes precedence.

To enable custom member formulas in a level, use its **Custom Members** property in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private). Clicking the edit (**...**) button beside this property's value displays the **Define Custom Member Column** dialog box in which you select or create a column to store the formulas. After you close this dialog box, if the dimension is write-enabled, you can create the formulas in the custom member formula pane of Dimension Editor (if the dimension is shared) or Dimension Browser. (To perform this action, make sure that the write-enabled dimension is included in a cube that was processed since the dimension last changed.) If the dimension is not write-enabled, you must use a tool other than Analysis Manager to insert the formulas into the column.

Note If a cube has a measure with its **Aggregate Function** property set to **Distinct Count**, adding a custom rollup operator or expression to a level will cause the cube's structure to become invalid.

See Also

Creating Custom Member Formulas

Properties Pane (Cube Editor Data View)

Using Custom Rollup Operators

Member Groups

A member group is a system-generated parent of a collection of consecutive dimension members. Member groups are created in a level that is added immediately above the level that contains the member groups' children. When end users browse a level that contains member groups, they see the names and cell values of the member groups. To end users, member groups look like ordinary members.

A level can contain either member groups or members. It cannot contain both.

Member groups rely on the **Order By** property of the next lower level. This property specifies the sort order of displayed members. Within this sort order, children of a member group are consecutive.

Member group names are created automatically. A member group name consists of the name of the first child of the member group followed by a hyphen (-) and the name of the last child of the member group.

There are two common uses of member groups. They can be used to provide an intermediate level for drilldown between a level with few members and one with numerous members. To do this, create a copy of the level that contains numerous members immediately above the original, and then create member groups in the new level. You can also use member groups to satisfy the maximum limit of 64,000 members under a single parent member. To do this, create a copy of the level that contains the excess members immediately above the original, and then create member groups in the new level. Each member group contains fewer than 64,000 children. For example, a Client dimension contains only an (All) level and a Client Name level containing 500,000 members. A copy of the Client Name level is created immediately above it and named Client Group. In the Client Group level, member groups are created. Each Client Name member now has a new parent in the Client Group level. If you want to hide the Client Group level, set its **Visible** property to **False**.

You can create member groups only in changing dimensions. You cannot create member groups in a dimension's top or bottom level. If this need arises, you can add a level such that the level in which you want to create member groups is no longer the top or bottom level. You can hide the added level by setting its **Visible** property to **False**. You cannot create member groups in two consecutive levels of a dimension.

Member groups are not supported for ROLAP dimensions.

To create member groups in a level, set its **Grouping** property to **Automatic** in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

When you change a level's **Grouping** property to **Automatic**, the following properties of the level are affected:

- **Member Keys Unique**: set to **True** for lowest level
- **Member Names Unique**: set to **False** and read-only
- **Order By**: set to **Name** and read-only
- **Member Key Column**: set to an empty string and read-only
- **Member Name Column**: set to an empty string and read-only

Also, if the dimension containing the level is a private dimension and its **Aggregation Usage** property is set to **Custom**, the **Aggregation Usage** property of the private dimension is set to **Standard**.

If the dimension table of a dimension that contains member groups is updated, and the dimension is subsequently processed, a new set of member groups is generated. The names and children of the new member groups are different from the old member groups.

See Also

Changing Dimensions

Creating Member Groups

Analysis Services

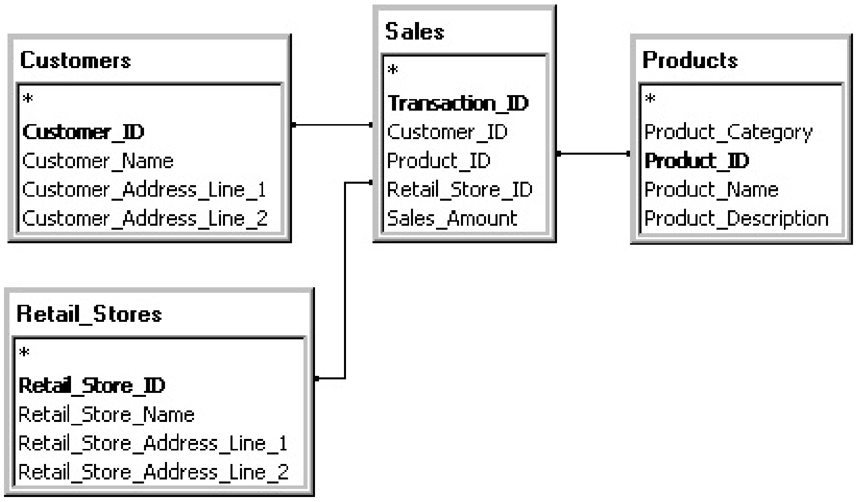
Measures

In a cube, a measure is a set of values that are based on a column in the cube's fact table and are usually numeric. In addition, measures are the central values of a cube that are analyzed. That is, measures are the numeric data of primary interest to end users browsing a cube. The measures you select depend on the types of information end users request. Some common measures are sales, cost, expenditures, and production count.

For each measure in a cube, the cube contains a value for every cell in the cube excluding the cells for the other measures. So, no matter which combination of members is used in a query, a measure value can be retrieved. The value may be retrieved from the cube's aggregations, its source data, a copy of it on the server, client cache, or a combination of these sources depending in part on the storage settings of the cube.

Measures are summarized by Microsoft® SQL Server™ 2000 Analysis Services, and the resulting aggregations are stored for quick retrieval by end users querying cubes. For more information, see Aggregations.

Consider a cube with the following schema and a single measure, Sales, based on the **Sales_Amount** column in the **Sales** fact table.



Assume a dimension for each of the other tables with a member for each customer, retail store, and product. If a query requests Sales for each customer, each retail store, and product A, each cell in the returned dataset contains a Sales value aggregated from the appropriate **Sales_Amount** values. For example, the Sales value in the cell for customer A, retail store A, and product A is produced by evaluating only the **Sales** table rows that contain the key values for all these members.

In the object hierarchy, measures are immediately subordinate to the cube. The measures of a cube are created when the cube is created. You select the measures for a regular cube when you build it with the Cube Wizard or Cube Editor. You also select measures when you build a virtual cube with the Virtual Cube Wizard. After a regular cube is built, you can maintain its measures in Cube Editor. After a virtual cube is built, you can maintain its measures in Virtual Cube Editor.

Each measure is derived from a column in a fact table. Because a regular cube can have only one fact table in its schema, all of the cube's measures must be contained within it.

After measures and their cube are created, partitions or aggregations are usually the next objects to be created. Partitions are created only if a cube is to contain multiple partitions; a single partition is created automatically for a cube when the cube is created. For more information, see Partitions and Aggregations.

In Analysis Manager, a measure is identified by the following icon.

Each measure specifies an aggregate function that determines how values in the measure's source column are aggregated. This function also determines how measure values for sibling members are aggregated to produce a value for their parent. The most commonly used aggregate function is **Sum**, but **Min**, **Max**, **Count**, and **Distinct Count** are also available. For more information, see Aggregate Functions.

Analysis Services supports measures based on both additive and nonadditive columns. Additive columns can be summed. For example, a monetary column is additive. Additive columns are suitable as measures in a cube regardless of the aggregate function that is used. Nonadditive columns cannot be summed meaningfully. For example, a numeric column containing an identifier such as **Account Number** is nonadditive. Nonadditive columns are also suitable as measures in a cube, but in order to be meaningful they must be summarized by the **Count** or **Distinct Count** aggregate function.

Note Using the **Distinct Count** aggregate function imposes restrictions on some cube functionality. For more information, see Using Aggregate Functions.

A measure can be derived from multiple columns combined in an expression. For example, the Profit measure is the difference of two numeric columns: Sales and Cost. For information about adding this type of measure to a cube, see Adding a Multiple-Column Measure to a Cube.

Calculated members can be used as measures. Calculated member values are created from formulas when the cube is browsed, but the values are not stored. Thus, calculated members save storage. For more information, see Calculated Members.

A cube contains a special type of dimension that contains a member for each measure. This dimension is called the Measures dimension. When end users browse the cube, they can slice by a member in the Measures dimension to display values for only a single measure, or they can place the Measures dimension on an axis so that they can see values for all the cube's measures. The Measures dimension is different from other dimensions insofar as it:

- Is created automatically when a cube is created.

- Cannot be displayed or edited in Dimension Editor. (Use Cube Editor or Virtual Cube Editor to maintain measures.)

- Is always flat (that is, always contains only one level).

For any dimension, including the Measures dimension, you can create a custom rule for dimension security to restrict end users' access to individual members. Because the Measures dimension is flat, many of the complexities of these custom rules regarding ancestors and descendants do not apply to the Measures dimension. For more information, see Custom Rules in Dimension Security.

Another method of restricting access to measures is to use cell security. For more information, see Cell Security.

If you are programming with Decision Support Objects (DSO), the class types associated with the measure are:

- **clsCubeMeasure**
- **clsPartitionMeasure**
- **clsAggregationMeasure**

How Measures Appear to End Users

Measures form the core of cube information presented to end users. Presentation may be tabular or graphical, depending on the client application with which end users browse cubes, but measures are the information end users focus on.

In tabular presentations, measures are displayed in rows and columns. Whereas a cube's dimensions determine the column and row headings, the measures are the data in the rows and columns. However, if you specify multiple measures in a cube, they too provide multiple headings to separate the measures.

In graphical presentations, measures may be displayed in a variety of ways, including lines, shapes, colors, shades, and shadows. Nevertheless, as in tabular presentations, the measures occupy the central, focal portion of the presentation while the dimensions provide peripheral labels.

See Also

clsAggregationMeasure

clsCubeMeasure

clsPartitionMeasure

Cube Editor - Schema View

Cube Wizard

Dimensions

Virtual Cube Editor

Virtual Cube Wizard

Aggregate Functions

Microsoft® SQL Server™ 2000 Analysis Services provides the following aggregate functions for use in measures.

Aggregate function	Returned value
Sum	The sum of the input values.
Min	The lowest of the input values.
Max	The highest of the input values.
Count	The number of input values.
Distinct Count	The number of unique input values.

A function is selected in the **Aggregate Function** property of a measure. This property is accessed in the properties pane of Cube Editor. For more information about how you can use these functions in real-world scenarios, see Using Aggregate Functions.

See Also

MDX Function List

Using Aggregate Functions

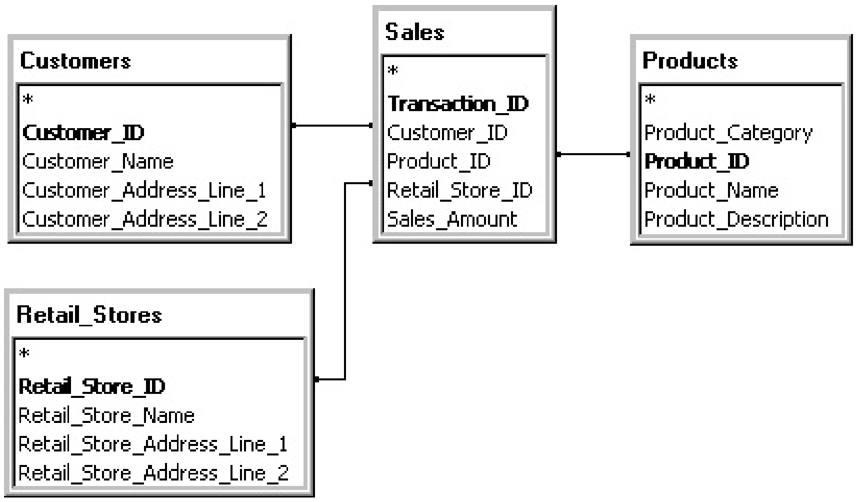
This topic contains examples for using the aggregate functions (**Sum**, **Min**, **Max**, **Count**, and **Distinct Count**) in measures. The examples for the query are based on the same cube cells as the following examples so that you can see the effects of changing the function.

The cube that these examples use has a single measure, Sales, based on the **Sales_Amount** column in the **Sales** fact table. The cube has three dimensions:

- Customers, based on the table **Customers** and containing these levels from highest to lowest:
    - (All)
    - Customer with **Customer_Name** as the member name column and **Customer_ID** as the member key column
- Retail Stores, based on the table **Retail_Stores** and containing these levels from highest to lowest:
    - (All)
    - Retail Store with **Retail_Store_Name** as the member name column and **Retail_Store_ID** as the member key column
- Products, based on the table **Products** and containing these levels from highest to lowest:
    - (All)
    - Product Category with **Product_Category** as the member name column and the member key column
    - Product with **Product_Name** as the member name column and **Product_ID** as the member key column

For more information about dimensions and levels, see Dimensions and Levels.

The cube's schema is shown here.



The cube's fact table, **Sales**, is shown here.

Transaction_ID	Customer_ID	Product_ID	Retail_Store_ID	Sales_ Amount
1	1	1	1	300
2	1	1	1	250
3	1	1	1	250
4	1	2	1	100
5	1	4	1	700
6	2	1	2	290
7	2	2	2	90
8	2	3	3	510
9	3	1	4	350
10	3	2	3	110
11	4	3	4	550
12	4	4	4	750

One of the cube's dimension tables, **Customers**, is shown here.

Customer_ID	Customer_Name	Customer_Address_Line_1	Customer_Address_Line_2
1	A	1 A Street	Aville, AA 55555
2	B	2 B Street	Bville, BB 55555
3	C	3 C Street	Cville, CC 55555
4	D	4 D Street	Dville, DD 55555

Another of the cube's dimension tables, **Retail_Stores**, is shown here.

Retail_Store_ID	Retail_Store_Name	Retail_Store_Address_Line_1	Retail_Store_Address_Line_2
1	A	1 A Avenue	Atown, AA 55555
2	B	2 B Avenue	Btown, BB 55555
3	C	3 C Avenue	Ctown, CC 55555
4	D	4 D Avenue	Dtown, DD 55555

The cube's final dimension table, **Products**, is shown here.

Product_ID	Product_Name	Product_Description	Product_Category
1	A	aaaa aaaa aaaa	AB
2	B	bbbb bbbb bbbb	AB
3	C	cccc cccc cccc	CD
4	D	dddd dddd dddd	CD

Sum

If a measure's **Aggregate Function** property value is **Sum**, the measure value for a cube cell is calculated by adding the values in the measure's source column from only the rows for the combination of members that defines the cell and the descendants of those members.

Examples

The following examples return values that represent accumulated Sales.

A: Querying One Atomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product A returns 800.

B: Querying One Nonatomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product category AB returns 900.

C: Querying Multiple Cube Cells

A query on the Sales measure places each retail store on the x-axis, nests products under product categories on the y-axis, and slices by All Customers. It returns the following dataset.

			All Retail Stores	A	B	C	D
All Products			4250	1600	380	620	1650
	AB		1740	900	380	110	350
		A	1440	800	290		350
		B	300	100	90	110	
	CD		2510	700		510	1300
		C	1060			510	550
		D	1450	700			750

Min

If a measure's **Aggregate Function** property value is **Min**, the measure value for a cube cell is calculated by taking the lowest value in the measure's source column from only the rows for the combination of members that defines the cell and the descendants of those members.

Examples

The following examples return values that represent the lowest Sales price.

A: Querying One Atomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product A returns 250.

B: Querying One Nonatomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product category AB returns 100.

C: Querying Multiple Cube Cells

A query on the Sales measure places each retail store on the x-axis, nests products under product categories on the y-axis, and slices by All Customers. It returns the following dataset.

			All Retail Stores	A	B	C	D
All Products			90	100	90	110	350
	AB		90	100	90	110	350
		A	250	250	290		350
		B	90	100	90	110	
	CD		510	700		510	550
		C	510			510	550
		D	700	700			750

Max

If a measure's **Aggregate Function** property value is **Max**, the measure value for a cube cell is calculated by taking the highest value in the measure's source column from only the rows for the combination of members that defines the cell and the descendants of those members.

Examples

The following examples return values that represent the highest Sales price.

A: Querying One Atomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product A returns 300.

B: Querying One Nonatomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product category AB returns 300.

C: Querying Multiple Cube Cells

A query on the Sales measure places each retail store on the x-axis, nests products under product categories on the y-axis, and slices by All Customers. It returns the following dataset.

			All Retail Stores	A	B	C	D
All Products			750	700	290	510	750
	AB		350	300	290	110	350
		A	350	300	290		350
		B	110	100	90	110	
	CD		750	700		510	750
		C	550			510	550
		D	750	700			750

Count

If a measure's **Aggregate Function** property value is **Count**, the measure value for a cube cell is calculated by adding the number of values in the measure's source column from only the rows for the combination of members that defines the cell and the descendants of those members.

Examples

The following examples return values that represent the number of Sales transactions.

A: Querying One Atomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product A returns 3.

B: Querying One Nonatomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product category AB returns 4.

C: Querying Multiple Cube Cells

A query on the Sales measure places each retail store on the x-axis, nests products under product categories on the y-axis, and slices by All Customers. It returns the following dataset.

			All Retail Stores	A	B	C	D
All Products			12	5	2	2	3
	AB		8	4	2	1	1
		A	5	3	1		1
		B	3	1	1	1	
	CD		4	1		1	2
		C	2			1	1
		D	2	1			1

Distinct Count

If a measure's **Aggregate Function** property value is **Distinct Count**, the measure value for a cube cell is calculated by adding the number of unique values in the measure's source column from only the rows for the combination of members that defines the cell and the descendants of those members.

A measure with an **Aggregate Function** property value of **Distinct Count** is called a distinct count measure. A distinct count measure can be used to count occurrences of a dimension's lowest-level members in the fact table. Because the count is distinct, if a member occurs multiple times, it is counted only once.

Distinct count measures are commonly used to determine for each member of a dimension how many distinct, lowest-level members of another dimension share rows in the fact table. For example, in a Sales cube, for each customer and customer group, how many distinct products were purchased? (That is, for each member of the Customers dimension, how many distinct, lowest-level members of the Products dimension share rows in the fact table?) Or, for example, in an Internet Site Visits cube, for each site visitor and site visitor group, how many distinct pages on the Internet site were visited? (That is, for each member of the Site Visitors dimension, how many distinct, lowest-level members of the Pages dimension share rows in the fact table?) In each of these examples, the second dimension's lowest-level members are counted by a distinct count measure.

This kind of analysis need not be limited to two dimensions. In fact, a distinct count measure can be separated and sliced by any combination of dimensions in the cube, including the dimension that contains the counted members.

A distinct count measure that counts members is based on a foreign key column in the fact table. (That is, the measure's **Source Column** property identifies this column.) This column joins the dimension table column that identifies the members counted by the distinct count measure.

Regular cubes are limited in their use of distinct count measures. Only one distinct count measure is allowed in a regular cube, and it is allowed only if the regular cube does not include a dimension with custom rollup operators or custom rollup formulas. Virtual cubes, however, do not share these limitations. A virtual cube can employ multiple distinct count measures and can also use custom rollup operators and custom rollup formulas in conjunction with distinct count measures.

Because distinct count measures are nonadditive, the presence of a distinct count measure significantly restricts the ability of Microsoft® SQL Server™ 2000 Analysis Services to preaggregate the cube. For this reason, it is recommended that each distinct count be placed in its own cube with no other measures. These cubes with distinct count measures can then be joined together with other cubes in a virtual cube that efficiently manages all of the measures.

Note If a cube uses a distinct count measure, the entire cube is treated as nonadditive. Nonadditive cubes do not support dynamically created members, therefore MDX functions, such as **VisualTotals**, which dynamically create members, will return an error if used on a nonadditive cube. This also affects other features, such as enabling visual totals in dimension security, which involve dynamically created members.

Examples

The following examples return values that represent the number of Sales transactions with a unique Sales price.

A: Querying One Atomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product A returns 2.

B: Querying One Nonatomic Cube Cell

A query on the Sales measure for customer A, retail store A, and product category AB returns 3.

C: Querying Multiple Cube Cells

A query on the Sales measure places each retail store on the x-axis, nests products under product categories on the y-axis, and slices by All Customers. It returns the following dataset.

			All Retail Stores	A	B	C	D
All Products			11	4	2	2	3
	AB		7	3	2	1	1
		A	4	2	1		1
		B	3	1	1	1	
	CD		4	1		1	2
		C	2			1	1
		D	2	1			1

See Also

MDX Function List

Measures

Virtual Cubes

Display Formats

You can select the format in which measure values are displayed to end users by using the **Display Format** property of the measure. This property is accessed in the properties pane of Cube Editor.

Although the properties pane provides a dropdown list for the **Display Format** property, you can specify many additional formats that are not in the list. You can specify named or user-defined formats that are valid in Microsoft® Visual Basic®. The following table contains some examples. They assume the regional setting in Control Panel on the client computer is English (United States).

Source data type	Named/user-defined format	Display Format value	Example output
Numeric	Named	General Number	123456789
			0
		Fixed	123456789.00
			0.00
	User-defined	$#,#.00	$123,456,789.00
			$.00
		#,#0.0000	123,456,789.0000
			0.0000
Date/time	Named	Medium Date	31-Dec-99
		Long Date	Friday, December 31, 1999
	User-defined	mm/dd/yyyy	12/31/1999
		mmm-dd-yyyy	Dec-31-1999
Boolean	Named	Yes/No	Yes
		True/False	True

For more information about valid named and user-defined formats, search on "format expression" in the Visual Basic section in the MSDN® Library at the Microsoft Web site.

Note In client applications, some display formats do not translate appropriately for all locales. If you are supporting multiple locales or a locale different than that of the Analysis server, you should test the display formats in client applications on computers set to those locales.

See Also

Cube Editor - Schema View

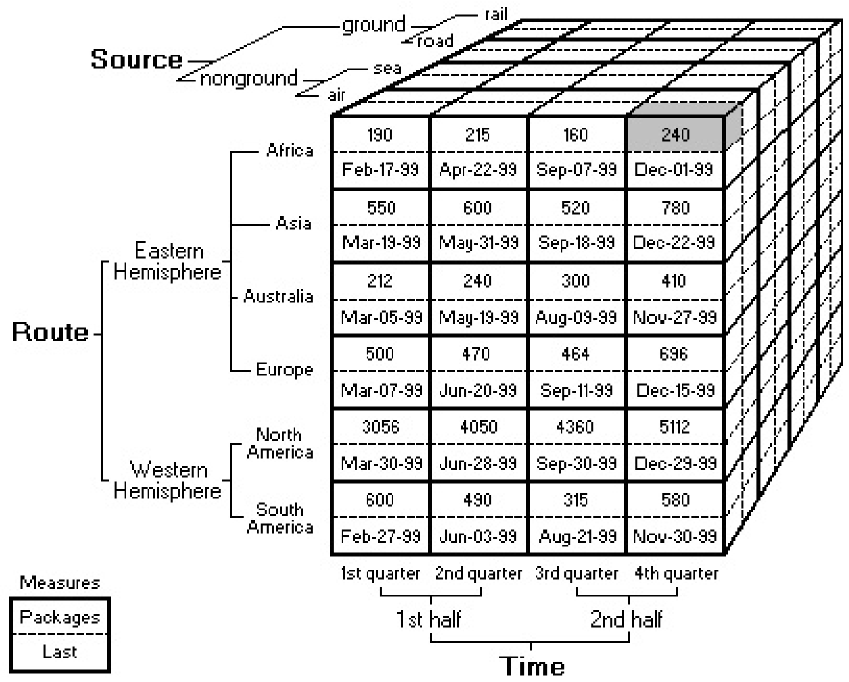
Using Cell Properties

Analysis Services

Cells

The cell is the atomic element of a cube, or the unique logical intersection of one member from every dimension associated with the cube. Essentially, a cube is composed of cells organized by measures, levels, and dimensions.

For example, the cube described by the following diagram has a single shaded cell.



The shaded cell is the intersection of the following members and dimensions:

- The air member of the Source dimension.
- The Africa member of the Route dimension.
- The 4th quarter member of the Time dimension.
- The Packages member of the Measures dimension.

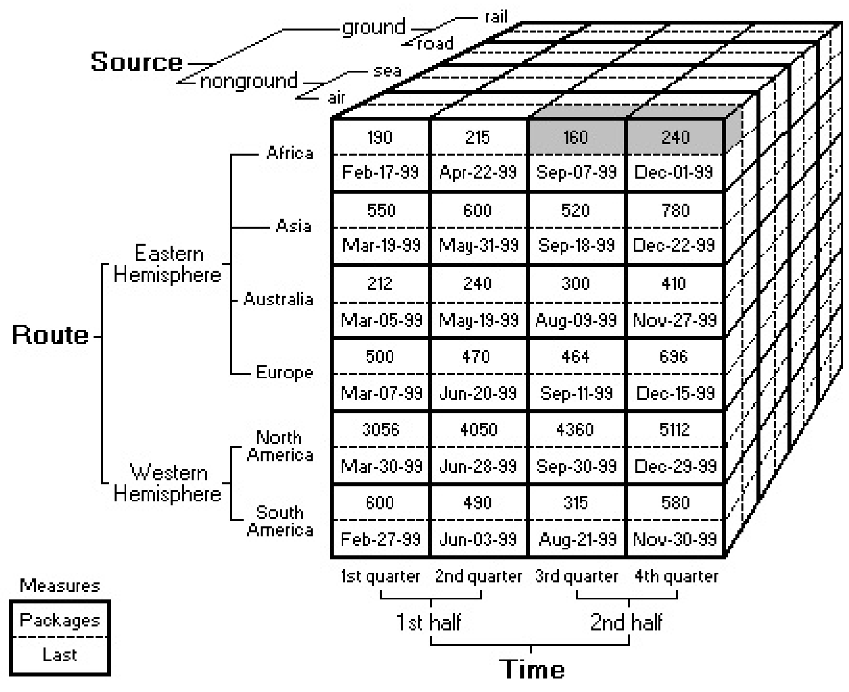
The value in the cell can be obtained from a number of sources. In the preceding example, the value in the cell is derived from the fact table of the cube, as all of the specified members are leaf members; a leaf member, hierarchically speaking, has no members below it.

However, a cell can be specified with nonleaf members; in this case, the value of the cell is derived from the aggregation of members associated with a nonleaf member.

For example, the intersection of the following members and dimensions refers to a cell whose value is supplied by aggregation:

- The air member of the Source dimension.
- The Africa member of the Route dimension.
- The 2nd half member of the Time dimension.
- The Packages member of the Measures dimension.

The 2nd half member of the Time dimension is a nonleaf member, so all of the members associated to it must be selected and aggregated, as shown in the following diagram.



Assuming the aggregations for the 3rd quarter and 4th quarter members are summations, the value of the specified cell is the total of all of the cells shaded in the preceding diagram.

Some nonleaf members, called data members, can have intrinsic data associated with them, as well as data derived from the aggregation of members below the nonleaf member. For more information about data members, see Data Members.

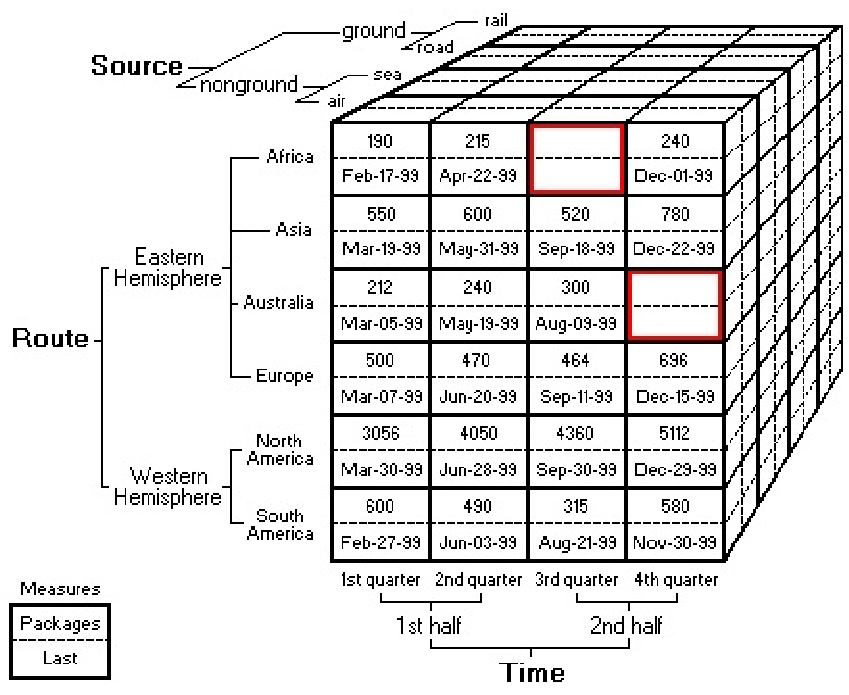
The cell values derived for members employed by custom members, custom rollups, and member groups are handled in a similar fashion. Cell values derived for calculated members, however, are based entirely on the Multidimensional Expressions (MDX) expression used to define the calculated member; there may be no actual cell data involved.

Additionally, the value of a cell can be derived by a formula executed for a specific set of cells, through the use of calculated cells. Calculated cells provide a great deal of flexibility in performing speculative analysis by supplying the ability to selectively apply formulas to specific cells in a section of the cube, or subcube, based on a condition the cells in the subcube must first fulfill.

Empty Cells

Not every cell in a cube must contain a value; there can be intersections in a cube that have no data. These intersections, called empty cells, occur frequently in cubes, because not every intersection of dimensions can contain a corresponding record in a fact table. The ratio of empty cells in a cube to the total number of cells in a cube is often referred to as the sparsity of a cube.

For example, the cube shown in the following diagram is similar to examples in this topic.



However, in this example, there were no air shipments to Africa for the third quarter or to Australia for the fourth quarter. Therefore, there is no data in the fact table to support the intersections of those dimensions and measures, so the cells at those intersections are empty.

In Microsoft® SQL Server™ 2000 Analysis Services, an empty cell is a cell with special qualities. Because empty cells can skew the results of crossjoins, counts, and so on, many MDX functions supply the ability to ignore empty cells for the purposes of the result generation.

For more information about empty cells, see Working with Empty Cells.

See Also

Calculated Cells

Measures

Dimensions

Levels

Custom Rollup Formulas and Custom Member Formulas

Member Groups

Cell Properties

A cell can have additional information that determines the content, font, and format of the data associated with it. This information can be used by client applications to present cell data.

The following table contains a list of supported cell properties with descriptions.

Property	Description
BACK_COLOR	The background color for displaying the **VALUE** or **FORMATTED_VALUE** property. For more information, see FORE_COLOR and BACK_COLOR Contents.
CELL_EVALUATION_LIST	A semicolon-delimited list of evaluated formulas applicable to the cell, in order from lowest to highest.
CELL_ORDINAL	The ordinal number of the cell in the dataset.
FORE_COLOR	The foreground color for displaying the VALUE or FORMATTED_VALUE property. For more information, see FORE_COLOR and BACK_COLOR Contents.
FONT_NAME	The font to be used to display the VALUE or FORMATTED_VALUE property.
FONT_SIZE	Font size to be used to display the VALUE or FORMATTED_VALUE property.
FONT_FLAGS	The bitmask detailing effects on the font. It can be the result of a bitwise **OR** operation of one or more of the following: • MDFF_BOLD

	• MDFF_ITALIC • MDFF_UNDERLINE • MDFF_STRIKEOUT
FORMAT_STRING	The format string used to create the FORMATTED_VALUE property value. For more information, see FORMAT_STRING Contents.
FORMATTED_VALUE	The character string that represents a formatted display of the VALUE property.
VALUE	The unformatted value of the cell.

In addition to the properties listed in the table, other providers may support provider-specific cell properties. The MDSCHEMA_PROPERTIES schema rowset contains information on supported cell properties, including data types and provider support.

For more information on supported cell properties and their usage, see Using Cell Properties.

See Also

MDSCHEMA_PROPERTIES

Calculated Cells

Calculated cells are cells whose value is calculated at run time using a Multidimensional Expressions (MDX) expression that you specify when you define the calculated cells. Additionally, the expression can be conditionally applied to specific cells, based on an MDX logical expression also specified when you define the calculated cells.

Calculated cells enable you to apply the functionality previously reserved for calculated members, custom members, and custom rollup formulas to a specific range of cells, or even to a single cell, allowing you to finely tune the performance of your cube or query.

Calculated cells consist of a multidimensional section of cells, defined by an MDX set expression, to which an MDX value expression is selectively applied depending upon a condition described by an MDX logical expression.

Calculated cells are constructed from three elements:

Calculation subcube

The MDX set expression used to define the slice of the cube upon which the calculated cells feature will work. The calculation subcube is defined by a list of single dimension sets, with each set containing one of the following:

- All members of a dimension (including the Measures dimension).

  Note This option typically excludes calculated members. To include calculated members, use an MDX expression that employs the **AllMembers** MDX function.

- A single specified member of a dimension (including the Measures dimension).

- All the members at a specified level in a dimension.

  Note This option typically excludes calculated members. To include calculated members, use an MDX expression that employs the

AllMembers MDX function.

- All the descendants of a specified member in a dimension.
- All the descendants of a specified member at a specified level in a dimension.
- An MDX expression that resolves into a set containing one of the above sets.

  The combination of this list of sets and every other member of all other dimensions not specified in the list of sets defines the calculation subcube.

Calculation condition

The MDX logical expression that further restricts the application of the calculated cells feature. The calculated cells condition expression is compared to each cell in the calculation subcube. If the logical expression evaluates to **True** for the cell, the calculated cells formula is applied and the cell returns the calculated value. If it evaluates to **False**, then the cell returns the original cell value.
The combination of the calculation subcube and the calculated cells condition is referred to as the calculation scope.

Calculation formula

The MDX value expression used to calculate the value of the cells contained in the calculation subcube.

This functionality sounds in many ways similar to calculated members, custom members, and custom rollup formulas, and indeed can be used in place of these features. But, calculated cells are much more than that.

For example, the following diagram depicts a cube in which there are three dimensions (not counting the Measures dimension).

Now, after reviewing this cube, you want to perform some speculation as to the quantity of packages shipped by air on the fourth quarter for the Western Hemisphere routes. You want to compute these cells as equivalent to 125% of the quantity of packages shipped by air on the third quarter for the same routes, but only if the quantity for the 4th quarter is less than 125% of the quantity of the third quarter.

You could create a custom member for the Air member of the Source dimension, but it would affect all of the cells for that member, including those intersecting all other members, as shown in the following diagram.

This custom member would, of course, affect all of the other routes and times. Another approach would be to create a custom member for the 4th quarter member of the Time dimension, but this would affect all of the other routes and sources, as shown in the following diagram.

You could also create custom members for each of the Western Hemisphere routes, but that would affect all of the sources and times, as shown in the next diagram.

Finally, you could combine all of the preceding examples, creating multiple custom members to cover all possible intersections, as shown in the following diagram, providing a complex MDX formula for each member to derive calculated values for just two cells.

None of these workarounds really provides the flexibility needed to work with only a specific slice of a cube, as all of these affect a complete member, not a specific range of cells. With calculated cells, you can define a specific range of cells, supply a condition for applying a formula, and apply a formula for that specific range of cells.

For the example detailed earlier in this topic, using this feature, you would first define the calculation subcube as shown in the following diagram.

Then you would create a calculation condition that would check each applicable cell to determine if it was less than 125% of the cell in the previous quarter. This would eliminate one of the cells, as shown in the following diagram.

The calculation formula would then be applied to the applicable cell, returning the value for the appropriate cells as shown in the following diagram.

To create calculated cells, use the Calculated Cells Wizard or Cube Editor. For more information about the Calculated Cells Wizard and Cube Editor, see Calculated Cells Wizard and Cube Editor - Schema View.

In Analysis Manager, the following icon identifies a calculated cells definition.

Calculated cells can be created with a global scope using the Calculated Cell Wizard or Cube Editor, available to all users who browse the cube in which the calculated cells are defined. However, calculated cells can also be created with a session scope, available only to MDX queries executed within the session in which the calculated cells are created. This feature allows greater flexibility for client applications.

See Also

Creating Calculated Cells

CREATE CELL CALCULATION Statement

Using WITH to Create Calculated Cells

Analysis Services

Cubes

A cube is a multidimensional structure that contains dimensions and measures. Dimensions define the structure of the cube, while measures provide the numerical values of interest to the end user. Cell positions in the cube are defined by the intersection of dimension members, and the measure values are aggregated to provide the values in the cells.

The following topics provide a basic conceptual overview of cubes.

Topic	Description
Introduction to Cubes	Introduces the concepts and behavior of cubes, including discussions on the structure, storage, and processing of cubes.
Cube Structure	Details the components of a cube, such as measures and dimensions, and discusses the importance of the cube schema.
Cube Storage	Describes the various techniques and storage modes, including the various uses of partitions, linked cubes, distributed partitioned cubes, and real-time cubes.
Cube Processing	Describes the nuances of processing cubes, including information on real-time OLAP.
Cube Varieties	Provides a brief overview of the varieties of cubes available in Microsoft® SQL Server™ 2000 Analysis Services.
Regular Cubes	Discusses regular cubes, which are based on tables. Regular cubes have their own aggregations, which utilize storage space. Regular cubes are distinct from linked cubes, virtual cubes, and local cubes.
Linked Cubes	Discusses linked cubes, which are based on regular cubes defined and stored on another Analysis server. A linked cube uses the aggregations of the cube on which it is based;

	therefore, it requires no storage space for aggregations.
Distributed Partitioned Cubes	Discusses distributed partitioned cubes, which are regular cubes with partitions stored on another Analysis server. A distributed partitioned cube uses aggregations that can be based on both local and remote Analysis servers, and thus can distribute processing across multiple Analysis servers.
Virtual Cubes	Discusses virtual cubes, which are logical cubes based on one or more regular cubes or linked cubes. Virtual cubes use the aggregations of their component regular cubes and the cubes on which their component linked cubes are based. Thus, virtual cubes require no storage space for aggregations.
Local Cubes	Discusses local cubes, which are contained in portable files and can be browsed without a connection to an Analysis server. Local cubes are based on tables. They do not have aggregations.
Real-Time Cubes	Discusses real-time cubes, which are regular cubes with dimensions and/or partitions enabled for real-time OLAP. The dimensions and/or partitions enabled for real-time OLAP and used by real-time cubes require no storage space for aggregations.
Write-Enabled Cubes	Discusses write-enabled cubes, in which authorized end users can update the cube's cell values.

Introduction to Cubes

Cubes are the main objects in online analytic processing (OLAP), a technology that provides fast access to data in a data warehouse. A cube is a set of data that is usually constructed from a subset of a data warehouse and is organized and summarized into a multidimensional structure defined by a set of dimensions and measures.

A cube provides an easy-to-use mechanism for querying data with quick and uniform response times. End users use client applications to connect to an Analysis server and query the cubes on the server. In most client applications, end users issue a query on a cube by manipulating the user interface controls, which determine the contents of the query. This spares end users from writing language-based queries. Precalculated summary data called aggregations provides the mechanism for rapid and uniform response times to queries. Aggregations are created for a cube before end users access it. The results of a query are retrieved from the aggregations, the cube's source data in the data warehouse, a copy of this data on the Analysis server, the client cache, or a combination of these sources. An Analysis server can support many different cubes, such as a cube for sales, a cube for inventory, a cube for customers, and so on.

Every cube has a schema, which is the set of joined tables in the data warehouse from which the cube draws its source data. The central table in the schema is the fact table, the source of the cube's measures. The other tables are dimension tables, the sources of the cube's dimensions. For more information about schemas, see Cube Structure.

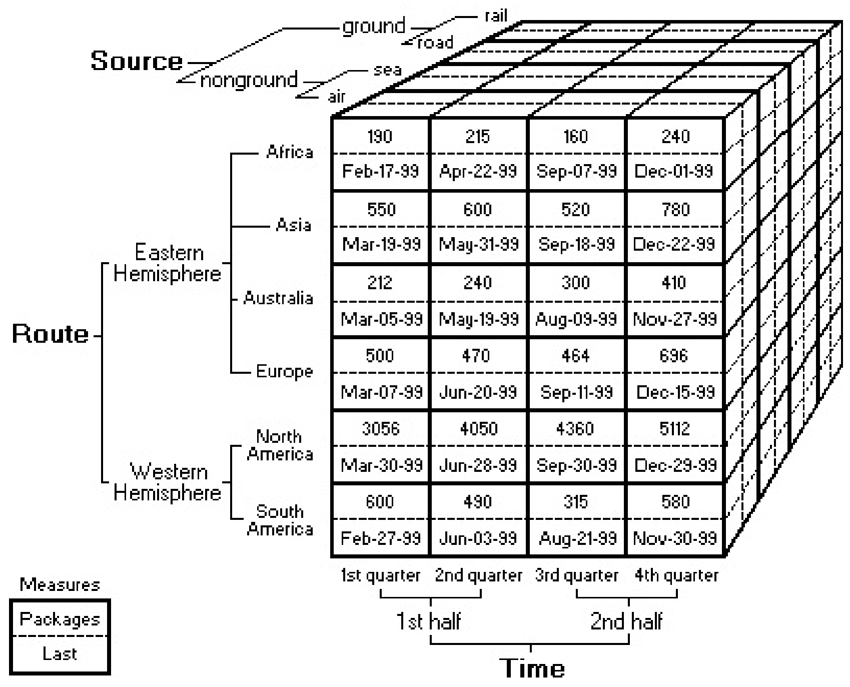
A cube is defined by the measures and dimensions that it contains. For example, a cube for sales analysis includes the measures Item_Sale_Price and Item_Cost and the dimensions Store_Location, Product_Line, and Fiscal_Year. This cube enables end users to separate Item_Sale_Price and Item_Cost into various categories by Store_Location, Product_Line, and Fiscal_Year.

Each cube dimension can contain a hierarchy of levels to specify the categorical breakdown available to end users. For example, the Store_Location dimension includes the level hierarchy: Continent, Country, Region, State_Province, City, Store_Number. Each level in a dimension is of finer granularity than its parent. For example, continents contain countries, and states or provinces contain cities. Similarly, the hierarchy of the Fiscal_Year dimension includes the levels Year, Quarter, Month, and Day.

Dimension levels are a powerful data modeling tool because they allow end users to ask questions at a high level and then expand a dimension hierarchy to reveal more detail. For example, an end user starts by asking to see Item_Cost values of products for the past three fiscal years. The end user may notice that 1998 Item_Cost values are higher than those in other years. Expanding the Fiscal_Year dimension to the Month level, the end user sees that Item_Cost values were especially high in the months January and August. The end user may then explore levels of the Store_Location dimension to see if a particular region contributed significantly to the high Item_Cost values, or may expand into the Product_Line dimension to see if Item_Cost values were high for a particular product group or product. This type of exploration, known as drilldown, is common in client applications.

For more information about dimensions, levels, and measures, see Dimensions, Levels, and Measures.

For example, consider the following Imports cube, which contains two measures, Packages and Last, and three dimensions, Route, Source, and Time.

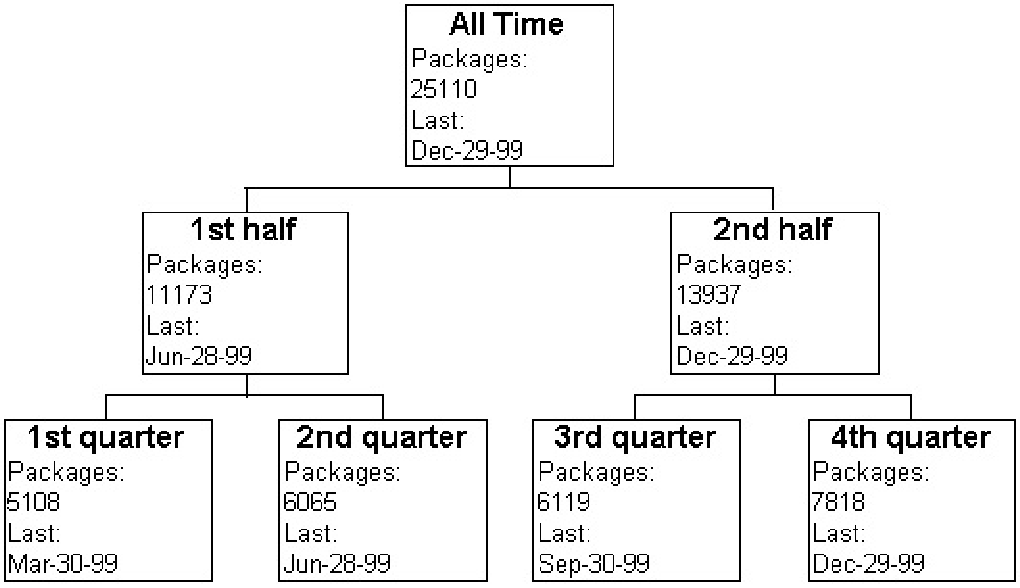


The smaller alphanumeric values around the cube are the members of the dimensions. Example members are ground, Africa, and 1st quarter.

The values within the cube represent the measures. Example measures are Packages: 190 and Last: Feb-17-99. These values exist for all cells in the cube but are shown only for those in the foreground. (In a real cube, the words *Packages* and *Last* would not appear in the cube cells, but they are shown here to distinguish the measures. In a real cube, measures are separated within a special dimension called the Measures dimension.)

The Packages measure represents the number of imported packages, and it aggregates by the **Sum** function. The Last measure represents the date of receipt, and it aggregates by the **Max** function. The Route dimension represents the means by which the imports reach their destination. The Source dimension represents the locations where the imports are produced. The Time dimension represents the quarters and halves of a single year.

End users of a cube can determine its measures' values for each member of every dimension. This is possible because measure values are aggregated by the members. For example, the measure values shown in the preceding illustration aggregate within the Time dimension as follows.



In addition to aggregating within a single dimension, measures aggregate for all combinations of members from different dimensions. This allows end users to evaluate measures by members in multiple dimensions simultaneously. For example, if an end user wants to analyze quarterly imports that arrived by air from the Eastern Hemisphere and Western Hemisphere, the end user can issue the appropriate query on the cube to retrieve the following dataset.

			Packages			**Last**		
			All Sources	**Eastern Hemisphere**	**Western Hemisphere**	**All Sources**	**Eastern Hemisphere**	**West Hem**
All Time			25110	6547	18563	Dec-29-99	Dec-22-99	Dec-
	1st half		11173	2977	8196	Jun-28-99	Jun-20-99	Jun-
		1st quarter	5108	1452	3656	Mar-30-99	Mar-19-99	Mar-
		2nd quarter	6065	1525	4540	Jun-28-99	Jun-20-99	Jun-
	2nd half		13937	3570	10367	Dec-29-99	Dec-22-99	Dec-
		3rd quarter	6119	1444	4675	Sep-30-99	Sep-18-99	Sep-
		4th quarter	7818	2126	5692	Dec-29-99	Dec-22-99	Dec-

A cube can contain up to 128 dimensions, each with thousands or millions of members, and up to 1,024 measures. A cube with a modest number of dimensions and measures usually satisfies the requirements of end users.

There are several varieties of cubes in Microsoft® SQL Server™ 2000 Analysis Services. Although regular cubes possess the characteristics of cubes described in this topic and its subtopics, other varieties of cubes do not share all of these characteristics. For more information about cube varieties, see Cube Varieties.

Cubes are immediately subordinate to the database in the object hierarchy. A database is a container for related cubes and the objects they share. You must create a database before you create a cube. For more information, see Databases.

In the object hierarchy, the following objects are immediately subordinate to the cube:

- Data sources

A cube has a single data source. It can be selected from the data sources in the database or created during cube creation. A cube's dimensions must have the same data source as the cube, but its partitions can have different data sources.

- Measures

  A cube's measures are not shared with other cubes. The measures are created when the cube is created. A cube can have up to 1,024 measures.

- Dimensions

  A cube's dimensions are either shared with other cubes in the database or private to the cube. Shared dimensions can be created before or during cube creation. Private dimensions are created when the cube is created. Although the term *cube* suggests three dimensions, a cube can have up to 128 dimensions.

- Partitions

  A single partition is automatically created for a cube when the cube is created. If you have installed Analysis Services for SQL Server 2000 Enterprise Edition, after creating a cube, you can create additional partitions in the cube.

- Cube roles

  Every cube must have at least one cube role in order to provide access to end users. Cube roles are derived from database roles, which can be created before or after cube creation. Cube roles are created after cube creation.

- Commands

  Commands are optional. Commands are created after cube creation.

After cubes are created, partitions or aggregations are usually the next objects to be created. For more information, see Partitions and Aggregations.

Creating a cube involves three steps:

Definition

The definition of a cube is based on the analytical requirements of end users. To define a cube, select a fact table and identify measures within the fact table. Then select or create dimensions, each composed of one or more columns from another table. The Cube Wizard provides an easy way to define cubes. Cube Editor offers additional flexibility for defining and modifying cube structures. For more information, see Building Cubes.

Aggregation design

After you define a new cube, you can design its aggregations using the Storage Design Wizard. Designing the aggregations specifies the summarization strategy. For more information, see Designing Storage Options and Aggregations.

Processing

After you design the aggregations of a new cube, process the cube with the **Full process** option. This action creates the aggregations. For more information about processing cubes with the **Full process** option, see Processing Cubes.

After you create a cube, use Cube Editor to maintain it.

If, after you process a cube, you change it, or its source data changes, it is usually necessary to process the cube again. Different processing options are appropriate in different circumstances. For more information about the processing options available for cubes, see Processing Cubes.

For information about the security options for cubes, see Cube Security.

If you are programming with Decision Support Objects (DSO), the class type associated with the cube is **clsCube**. For more information, see clsCube.

See Also

Cube Editor - Schema View

Measures

Dimensions

Levels and Members

Data Sources

Partitions

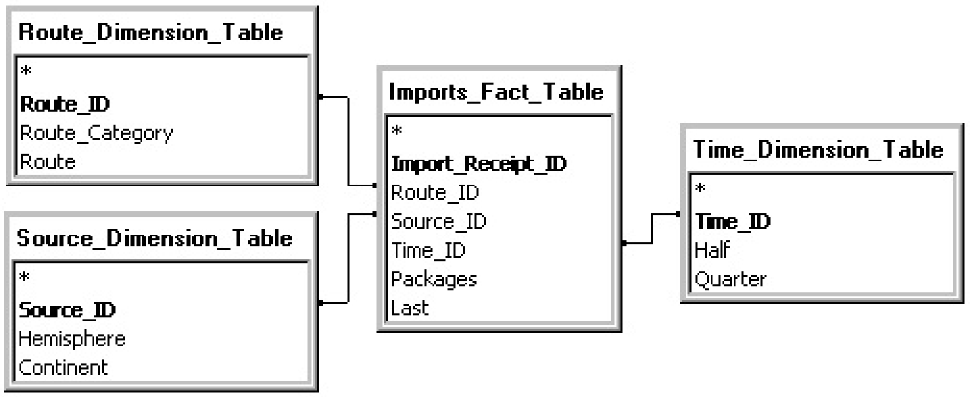
Roles

Commands

Cube Structure

A cube's structure is defined by its measures and dimensions. They are derived from tables in the cube's data source. The set of tables from which a cube's measures and dimensions are derived is called the cube's schema. Every cube schema consists of a single fact table and one or more dimension tables. The cube's measures are derived from columns in the fact table. The cube's dimensions are derived from columns in the dimension tables.

For example, a cube has the following schema.



The cube's measures and dimension levels are derived from the following columns.

Measure or level	Members	Source table	Source column	Sample column value
Packages measure	Not applicable	**Imports_ Fact_Table**	**Packages**	12
Last measure	Not applicable	**Imports_ Fact_Table**	**Last**	May-03-99
Route Category level in Route dimension	nonground, ground	**Route_ Dimension_ Table**	**Route_ Category**	nonground
Route level in	air,	**Route_**	**Route**	sea

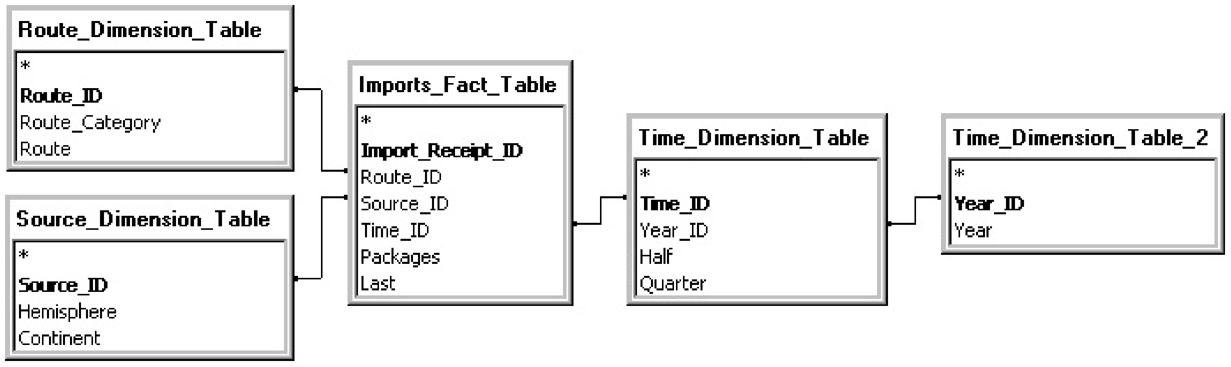
Route dimension	sea, road, rail	**Dimension_ Table**		
Hemisphere level in Source dimension	Eastern Hemisphere, Western Hemisphere	**Source_ Dimension_ Table**	**Hemisphere**	Eastern Hemisphere
Continent level in Source dimension	Africa, Asia, Australia Europe, N. America, S. America	**Source_ Dimension_ Table**	**Continent**	Europe
Half level in Time dimension	1st half, 2nd half	**Time_ Dimension_ Table**	**Half**	2nd half
Quarter level in Time dimension	1st quarter, 2nd quarter, 3rd quarter, 4th quarter	**Time_ Dimension_ Table**	**Quarter**	3rd quarter

A single cube cell is usually derived from multiple rows in the fact table. For example, the cell in the Imports cube for the air member, the Africa member, and the 1st quarter member is derived from the following rows in the **Imports_Fact_Table**.

Import_ Receipt_ID	**Route_ID**	**Source_ID**	**Time_ID**	**Packages**	**Last**
3516987	1	6	1	15	Jan-10-99
3554790	1	6	1	40	Jan-19-99
3572673	1	6	1	34	Jan-27-99
3600974	1	6	1	45	Feb-02-99
3645541	1	6	1	20	Feb-09-99
3674906	1	6	1	36	Feb-17-99

In the preceding table, the fact that each row has the same values for **Route_ID**, **Source_ID**, and **Time_ID** indicates that these rows contribute to the same cube cell.

There are two common types of cube schemas: star and snowflake. In a star schema, each dimension table joins to the fact table. The Imports cube schema shown earlier in this topic is a star schema. In a snowflake schema, one or more dimension tables join to another dimension table rather than to the fact table. The dimension tables that do not join to the fact table are for dimensions with multiple dimension tables. For example, the administrator of the Imports cube wants to expand the cube to multiple years. To accomplish this, the administrator adds a **Year_ID** column to the **Time_Dimension_Table**, the **Time_Dimension_Table_2** to the cube's schema, and a Year level to the Time dimension. (The Year level is based on the **Time_Dimension_Table_2.Year** column.) After these changes are made, the Time dimension is based on two dimension tables, and the cube's schema is snowflake, as shown in the following illustration.



After you change a cube's structure, usually you must process the cube with the **Full process** option so that changes can be seen by end users. For more information, see Processing Cubes.

See Also

Dimensions

Measures

Cube Storage

Cube data and aggregations can be stored with different techniques and in a variety of modes.

Cubes can require substantial storage to contain the data and aggregations in multidimensional structures. One factor that affects storage requirements is sparsity. For example, if one dimension contains sales representatives and another dimension contains regions, cells at the intersection of the Northeast sales representative and the Southwest region will probably be empty.

Microsoft® SQL Server™ 2000 Analysis Services uses several techniques for minimizing cube storage requirements:

- Storage is not allocated for empty cells, which compensates for performance issues that can be caused by sparsity.
- Data compression is employed.
- A sophisticated algorithm designs efficient summary aggregations to minimize storage without sacrificing speed.

Storage options enable you to create an OLAP storage strategy tailored to your needs by selecting appropriate storage modes and locations for cube data.

For more information, see Partition Storage and Aggregations.

Partitions

Cubes can be divided into partitions, and each partition can be stored using a different storage mode. For example, you can create a cube containing data for several years of transactions and partition it at year boundaries. You can store the partition for the current year in a multidimensional OLAP (MOLAP) structure with a high percentage of aggregations for quick response to users. You can use hybrid OLAP (HOLAP) to store the partition for the previous year, providing good response to summary queries with reduced storage needs. You can store data for years prior to the previous year in one or more relational OLAP (ROLAP) partitions with a smaller percentage of aggregations, saving on storage space with a tradeoff in query response.

Caution Partitioning cubes and merging partitions are advanced techniques. It is possible to create partitioned cubes that contain incorrect data. For more information about specific precautions, see Fact Table Considerations When Merging Partitions and Merging Partitions That Have Data Slices.

The partitions of a cube are not visible to the end user. In the preceding example, any query valid for the entire cube will execute, but queries that return older data will take more time than those that request newer information.

Analysis Services provides the Partition Wizard to assist in creating partitions. However, it is important that partitions be defined to contain mutually exclusive data. A cube may return incorrect results for some queries if a portion of the cube's data is included in more than one of its partitions.

Partitions can be stored on different Analysis servers, providing a clustered approach to cube storage and distributing workload across Analysis servers. For more information, see Distributed Partitioned Cubes and Remote Partitions.

Two partitions of a cube can be merged into a single partition, which can then be merged into another partition, and so on until only a single partition remains. For example, four partitions, each containing data for a quarter, can be merged into a single partition that contains the data for the entire year. There are precautions that need to be considered when merging partitions to ensure the resulting partition contains correct data.

Linked Cubes

The storage requirements of multiple copies of a cube on different Analysis servers can be greatly reduced by replacing the copies with linked cubes. A linked cube is based on a cube on another Analysis server, referred to as the source cube. A linked cube uses the aggregations of its source cube and has no data storage requirements of its own. Therefore, by maintaining a single source cube on one Analysis server and creating linked cubes on the other Analysis servers that require the cube, you can save a large amount of storage resources. For more information, see Linked Cubes.

Real-Time Cubes

Real-time cubes offer the opportunity of using relational OLAP (ROLAP) dimensions and partitions enabled for real-time OLAP. A ROLAP partition can be used to handle rapidly changing fact table data, eliminating the need for frequent reprocessing of a cube and the resulting inconvenience and unavailability. In this case, a ROLAP partition enabled for real-time OLAP requires no additional storage space, because all aggregations are performed as needed. Unlike ROLAP dimensions, ROLAP partitions must meet special requirements if they are to be used with real-time OLAP. ROLAP partitions must either have no stored aggregations or be based on an indexed view. For more information about these requirements, see Real-Time Cubes.

See Also

Merging Partitions

Remote Partitions

Cube Processing

After making structural changes to a cube, you must process the cube before attempting to browse its data. Process your cube after completing any of the following:

- Building the cube and designing its storage options and aggregations
- Changing the cube's structure (measures, dimensions, and so on) and saving the changes to the cube
- Changing the structure of a shared dimension used in the cube

Also, if data in the cube's data warehouse has been added or changed, processing is recommended in order to ensure accurate results when browsing the cube.

When you process a cube, the aggregations designed for the cube are calculated and the cube is loaded with the calculated aggregations and data. Processing a cube involves reading the dimension tables to populate the levels with members from the actual data, reading the fact table, calculating specified aggregations, and storing the results in the cube. After a cube is processed, users can query it.

There are three ways to process a cube. If you are modifying the structure of the cube, you may be required to process the cube with the **Full Process** option. If you are adding new data to the cube, you can process the cube with the **Incremental update** option. To clear out and replace a cube's source data, you can use the **Refresh data** processing option.

In addition to these three mutually exclusive options, a fourth option can be selected in conjunction with any of these options. This option allows you to incrementally update the cube's dimensions as part of the cube processing. This option is called **incrementally update the dimensions of this cube**.

These options are available in the **Process a Cube** dialog box, which is displayed when you right-click a cube in the Analysis Manager tree pane and then click **Process**.

Real-Time OLAP

Processing a cube can take a great deal of time, especially for complex cubes with dimensions containing millions of members. Frequently processing cubes based on rapidly changing data, such as data found in online transaction processing (OLTP) databases, can be a difficult task. Very often such cubes contain stale data, with aggregations based on a view of the data that is no longer valid.

Real-time OLAP provides a way to automatically process the relational OLAP (ROLAP) dimensions and/or partitions based on Microsoft® SQL Server™ 2000 relational tables when changes to the underlying dimension or fact tables occur. This allows real-time cubes based on rapidly changing data to be automatically updated and always available to end users. For more information about real-time cubes, see Real-Time Cubes.

See Also

Processing Cubes

Updating and Refreshing Cube Data

Cube Varieties

Analysis Services provides several varieties of cubes. All cubes, with the exception of local cubes, are based in part on a regular cube. A local cube can be based on a regular cube, but does not require a regular cube in order to be created.

The following table displays the features supported among the cube varieties.

	Regular	**Virtual**	**Linked**	**Offline**	**Write-enabled**	**Distributed Partitioned**	**Real-time**
Regular	----	N	N	N	Y	Y	Y
Virtual	Y**	N	Y**	N	N	N	Y*
Linked	Y***	Y***	N	N	N	N	Y
Offline	N	N	N	----	N	N	N
Write-enabled	Y	N	N	N	----	Y	Y
Distributed Partitioned	Y	N	N	N	Y	----	Y
Real-time	Y	Y*	Y	N	Y	Y	----

* Only if based on a source real-time cube; virtual cubes do not directly support real-time updates.
** Virtual cubes can be based only on regular or linked source cubes.
*** Linked cubes can be based only on regular or virtual source cubes.

Regular Cubes

A regular cube is a cube that is not a linked cube, virtual cube, or local cube. (Every cube is regular, linked, virtual, or local.)

A regular cube is based on tables in the databases specified in the data sources of the cube's partitions. In contrast, linked cubes and virtual cubes are based on other cubes.

A regular cube must contain at least one partition. A regular cube can contain multiple partitions provided that you have installed Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition. Linked cubes, virtual cubes, and local cubes do not contain partitions.

Regular cubes have aggregations and require storage space to store them. In contrast, linked, virtual, and local cubes do not have aggregations. Linked and virtual cubes use the aggregations of the cubes on which they are based. Aggregation data for a regular cube is stored in subfolders of the Data folder of the Analysis server on which the cube is defined, or in subfolders of the Data folder of another Analysis server, or in the databases specified in the data sources of the cube's partitions, or in a combination of these locations, depending on the storage modes and types of the cube's partitions. For more information, see Partition Storage and Remote Partitions.

If a regular cube contains partitions with a storage mode of multidimensional OLAP (MOLAP), the cube also requires storage space for the copies of the source data for those partitions. In contrast, linked and virtual cubes do not require storage space for this purpose. However, a local cube with a storage mode of MOLAP contains a compressed copy of the local cube's fact table data.

A regular cube is the only variety of cube that can be write-enabled. For more information, see Write-Enabled Cubes.

In Analysis Manager, a regular cube is identified by the following icon.

🟨

See Also

Aggregations

Introduction to Cubes

Linked Cubes

Local Cubes

Partitions

Virtual Cubes

Linked Cubes

A linked cube is based on another cube that is defined and stored on another Analysis server. To end users, linked cubes appear and function like regular cubes. By using linked cubes, you can create, store, and maintain a cube on one Analysis server while the cube is also available as linked cubes on multiple Analysis servers. This arrangement can provide many benefits, including:

- Less overall storage and maintenance is required, compared to storing and maintaining multiple copies of a cube on multiple Analysis servers.
- One organizational unit can maintain ownership of a cube and exclusive update rights but make the cube available to other units as linked cubes.
- Sensitive information can be stored in data sources and cubes on secured server computers but made available through other server computers as linked cubes.

Note You can create linked cubes only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

Certain terms are associated specifically with linked cubes. The cube on which a linked cube is based is the source cube. The Analysis server that stores the source cube is the publishing server. An Analysis server that stores a linked cube is a subscribing server.

The source cube of a linked cube can be a regular or virtual cube.

When you create a linked cube, do not allow the publishing server and subscribing server to be the same server.

A linked cube always has a ROLAP storage mode and has no aggregations. A linked cube uses the aggregations of its source cube.

The following security requirements apply to linked cubes:

- To create a linked cube, you must have access to the source cube. This access can be provided by including your username in either of the following on the publishing server: the OLAP Administrators group or a cube role for the source cube. If the account does not have access to the source cube, processing the linked cube fails.

- If a linked cube is defined on an Analysis server and Microsoft Windows NT® Integrated Security is used for user authentication, the Analysis server service (MSSQLServerOLAPService) logon account can be either a domain user account or a system account having access to the source cube. However, if a system account is used to create a linked cube, you can still create the linked cube but you will receive a warning if you are not using HTTP or secure HTTP (HTTPS) authentication and the Security Support Provider Interface (SSPI) property in the data source connection string is empty.

- You can also use a data source administered by a Security Support Provider Interface (SSPI) provider such as Negotiate, Kerberos, or NTLM to create a linked cube as long as the account credentials are recognized by the publishing server.

- Data sources using HTTP or secure HTTP authentication are also supported in relation to creating linked cubes.

- If cell security is enforced on a publishing server for a cube, defining a link to this cube is not possible.

In Analysis Manager, a linked cube is identified by the following icon.

In some cases, the dimensions in a linked cube might not be suitable for its end users, but there are features, such as member properties of a dimension, that you can use to resolve this problem. For example, the source cube contains dimensions created for English speakers, but the linked cube is used by French speakers. Or, for example, the source cube contains a Calendar dimension created for universities with four quarters per year, but the linked cube is used by end users at universities with two semesters per year. In cases of different languages among the publishing server and the subscribing servers, you can create member properties on the subscribing server that replace member captions based on the locale ID of the end user's computer. By adding these member properties to a dimension and columns to its table, the dimension can support multiple languages. For more information, see Multiple Language Implementation Using Member Properties.

Before designing linked cubes, consider the following:

- Writeback is not supported in linked cubes.
- Linked cubes cannot be created from source cubes that include shared or private ROLAP dimensions.
- If structural changes, such as adding or deleting measures or dimensions, have been made to a source cube, the linked cube must be reprocessed so that the changes are visible to end users.
- DLLs that contain user-defined functions registered on a publishing server and referenced by the source cube must also be registered on subscribing servers; otherwise they cannot be referenced by the linked cube. Due to maintenance issues, it is not recommended for published cubes to use user-defined functions.
- If a source cube has members with custom member formulas or custom rollup formulas that reference other cubes on the publishing server, then the cube that is referenced on the publishing server must be referenced on the subscribing server. You can make the reference on the subscribing server by creating another linked cube that is based on the cube being referenced in the formulas. This linked cube must have the same name on the subscribing server as the source cube on the publishing server.

See Also

Building a Linked Cube

Managing Linked Cubes

Distributed Partitioned Cubes

Distributed partitioned cubes are regular cubes that employ partitions on multiple Analysis servers. Partitions stored on Analysis servers other than the Analysis server which stores the meta data for the partition are referred to as remote partitions and are very useful when dealing with complex cubes based on dimensions having millions of members.

Distributed partitioned cubes enable the processing of queries to be distributed across multiple Analysis servers, because the processing for each partition is performed on the Analysis server storing the partition data. In this way, performance can be increased when querying and processing distributed partitioned cubes. However, the performance increase can be impacted by network speed because information is passed from the remote Analysis server to the Analysis server processing the distributed partitioned cube.

As with other cubes, a distributed partitioned cube must be processed before client applications can browse it. Processing a distributed partitioned cube establishes the links to the remote Analysis servers containing the data for the remote partitions. The meta data for a remote partition can be viewed and updated only on the Analysis server where the partition and its parent distributed partitioned cube are defined. It cannot be viewed or updated on the remote Analysis server. Thus, a cube that contains remote partitions can be administered along with its partitions on a single Analysis server.

See Also

Remote Partitions

Partition Wizard

Virtual Cubes

A virtual cube is a combination of multiple cubes in one logical cube, somewhat like a relational database view that combines other views and tables. When you create a virtual cube, you select measures and dimensions from the consolidated set of dimensions and measures in the underlying component cubes. End users see the virtual cube as a single cube.

A virtual cube can also be based on a single cube to expose only selected subsets of its measures and dimensions.

A virtual cube can include normal or linked cubes as component cubes.

Because virtual cubes store only their definitions and not the data of their component cubes, they require virtually no physical storage space. You can use virtual cubes to create combinations and variants of existing cubes without using significant additional storage.

A virtual cube can provide a valuable security function by limiting the access of some users when viewing the underlying cubes. If some of a cube's information is sensitive and not suitable for all users, you can create a virtual cube from the existing cube and omit the sensitive information. Then create two security roles: the first containing the users permitted to see the sensitive information, and the second containing the other users. Finally, grant the first role access to the cube and the second role access to the virtual cube.

Virtual cubes also are useful in relation to working with OLAP data mining models. For example, when creating an OLAP mining model, you can also create a dimension to store the results of the mining model analysis and a virtual cube that contains this dimension and the mining model's source cube.

In Analysis Manager, a virtual cube is identified by the following icon.

After you create a virtual cube, you must process it before client applications can browse it. Processing a virtual cube establishes the internal links to the specified dimensions and measures in its underlying cube or cubes. This linking operation is performed quickly. However, processing a virtual cube automatically triggers processing of all underlying cubes that need to be processed, which can add significant time.

See Also

Building a Virtual Cube

Data Mining Models

Mining Model Wizard

Local Cubes

Local cubes are appropriate in situations in which an Analysis server is unavailable. A local cube is stored in a single, portable file that can be stored on both server and nonserver computers. End users can browse local cubes without a connection to an Analysis server. Local cubes are the only variety of cube that provides this capability.

Local cubes are based on tables. Local cubes do not have aggregations or contain partitions.

The storage mode of a local cube can be either multidimensional OLAP (MOLAP) or relational OLAP (ROLAP). If the storage mode is MOLAP, end users can browse the local cube without a connection to its source data.

Unlike other varieties of cubes, a local cube is not an object in the object hierarchy shown in the Analysis Manager tree pane.

After a local cube is created, if its source data changes, the local cube can be refreshed to incorporate the new version of the source data.

You can use Microsoft® Excel 2000 to create, refresh, and browse local cubes. In Excel, local cube files are called offline cube files. For more information, see the Excel documentation.

By default the file extension associated with local cube files is .cub.

See Also

Building Local Cubes

Object Architecture Overview

PivotTable Service

Real-Time Cubes

A real-time cube is a regular cube that employs relational OLAP (ROLAP) partitions or dimensions that support the real-time OLAP feature of Microsoft® SQL Server™ 2000 Analysis Services.

With regular cubes, each time the data supporting the dimensions or partitions of a cube changes, the cube must be reprocessed to recreate its aggregations. This can lock the cube for long periods of time, preventing users from querying the cube. For fact table data that changes often, this approach is impractical. Often, workarounds to allow rapid cube updates are employed, such as designing a cube with zero aggregations and writing applications employing Decision Support Objects (DSO) code to suspend and resume the cube whenever underlying fact table data changes. However, these workarounds are not efficient or practical.

Real-time OLAP resolves this issue by enabling ROLAP dimensions and partitions to automatically refresh themselves when data in their underlying dimension or fact tables changes. When working in concert with SQL Server 2000 as the relational data source, Analysis Services can periodically poll the data source for notifications about updates to dimension or fact tables associated with specific ROLAP dimensions or partitions enabled for real-time updates. If the Analysis server finds that a change to a dimension or fact table has occurred, it can respond to the notifications by flushing the Analysis server cache and automatically reprocessing the associated ROLAP dimensions or partitions.

The Analysis server maintains a "listener" thread that supports the notification process for all ROLAP dimensions and ROLAP partitions using a SQL Server 2000 data source, for each distinct SQL Server 2000 server. The Analysis server monitors the tables specified in the FROM clause of the ROLAP dimension or partition, and reprocesses the dimension or partition whenever a notification about one of the monitored tables is received from SQL Server 2000.

The effect of real-time OLAP on client applications is to allow a real-time view of OLAP data, instead of having to reprocess a cube every time data in the dimension or fact tables associated with the dimensions or partitions of a cube changes, as with regular cubes. Real-time OLAP expands the ability of Analysis

Services to construct cubes based on rapidly changing data, including the ability to support real-time cubes directly based on OLTP data, without the maintenance and administrative difficulties such cubes would have caused in earlier releases of Analysis Services.

While ROLAP dimensions can easily support real-time updates, ROLAP partitions must meet certain qualifications in order to support this new functionality. They must either have zero aggregations or be based on SQL Server 2000 indexed views, and they cannot be remote partitions. For more information on the requirements of indexed views for ROLAP partitions, see Indexed Views for ROLAP Partitions.

Real-time OLAP can be enabled by using either Dimension Editor or the Dimension Wizard to enable real-time updates on ROLAP dimensions, or by using the Storage Design Wizard to enable real-time updates on ROLAP partitions that meet the qualifications discussed earlier in this topic.

See Also

Dimension Editor - Schema View

Dimension Wizard

Storage Design Wizard

Write-Enabled Cubes

If you write-enable a cube, client applications can record changes to the cube's data. These changes, known as writeback data, are stored in a table separate from the cube and its underlying data, but they are incorporated into query results as if they are part of the cube data. Write-enabled cubes allow end users to explore scenarios by changing cell values and analyzing the effects of the changes on cube data.

An end user's change is stored in the writeback table as a difference from the currently displayed value. For example, if an end user changes a cell value from 90 to 100, the value +10 is stored in the writeback table, along with the time of the change and information about the end user who made it. The net effect of accumulated changes is displayed to client applications. The original value in the cube is preserved, and an audit trail of changes is recorded in the writeback table.

Changes to atomic and nonatomic cube cells are handled differently. (An atomic cell represents a lowest-level member of every dimension in the cube. The value of an atomic cell cannot be reduced by drilling down or slicing.) If a cube is write-enabled, changes can be made to an atomic cell. Changes can be made to a nonatomic cell only if the client application provides a means of distributing the changes among the atomic cells that make up the nonatomic cell. Programmers of client applications can use the UPDATE CUBE statement to distribute changes made to nonatomic cells. For more information, see UPDATE CUBE Statement.

Regardless of whether a client application distributes changes made to nonatomic cells, when queries are evaluated, changes in the writeback table are applied to nonatomic as well as atomic cell values so end users can view the effects of the changes throughout the cube.

End user changes are kept in a separate writeback table that you can:

- Convert to a partition to permanently incorporate changes into the cube. This action makes the cube read-only. You can specify a filter expression to select the changes you want to convert.

Note Converting to a partition is available only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

- Discard to return the cube to its original state. This action makes the cube read-only.

For more information about write-enabling a cube, converting writeback tables to partitions, and deleting writeback data, see Maintaining Write-Enabled Cubes and Writeback Data.

An end user is permitted to record changes in a cube's writeback table only if the end user belongs to a cube role with read/write access to the cube's cells. For each cube role, you can control which cube cells can and cannot be updated. For more information, see Cell Security.

Write-enabled cubes and write-enabled dimensions are different but complementary features. A write-enabled cube gives users the ability to update cube cells, whereas a write-enabled dimension gives users the ability to update members. Using these two features in combination is optional. For example, a write-enabled cube does not have to include any write-enabled dimensions. Different procedures are used to write-enable cubes and dimensions and to maintain their security. For more information about write-enabled dimensions, see Write-Enabled Dimensions.

If you want to write-enable a cube with a Microsoft Access database as a data source, do not use Microsoft OLE DB Provider for ODBC Drivers in the data source definitions for the cube, its partitions, or its dimensions. Instead, you can use Microsoft Jet 4.0 (or later) OLE DB Provider.

A cube can be write-enabled only if all of its measures use the **Sum** aggregate function.

Note You cannot use Microsoft SQL Server 2000 Analysis Services to process a write-enabled cube created in Microsoft SQL Server 7.0 OLAP Services if any of the measures of the write-enabled cube use an aggregate function other than **Sum**.

Linked cubes and local cubes cannot be write-enabled. A virtual cube cannot be write-enabled; however, if one or more of its component cubes is write-enabled, virtual cube cells derived solely from the write-enabled cubes can be updated.

Analysis Services

Partitions

Partitions are used to store and manage precalculated aggregations and, sometimes, source data. They also offer flexibility in storing such data in multiple locations and in optimizing its access; they are also useful in managing the growth of cubes over time.

The following topics describe structural considerations for partitions.

Topic	Description
Introduction to Partitions	Describes basic characteristics of partitions and provides an example of a multiple-partition cube
Partition Structure	Identifies requirements pertaining to the structure of partitions
Partition Storage	Describes differences between the three storage modes
Indexed Views for ROLAP Partitions	Identifies conditions necessary for Microsoft® SQL Server™ 2000 Analysis Services to create indexed views for aggregations
Remote Partitions	Describes remote partitions and their restrictions
Incremental Updates and Partitions	Identifies considerations relative to updating partitions incrementally

Introduction to Partitions

Partitions allow the source data and aggregate data of a cube to be distributed among multiple server computers. Each partition in a cube can have a different data source. These data sources can reference relational databases on various computers. In addition, aggregate data of each partition can be stored on the Analysis server computer where the partition is defined, on another Analysis server computer, or in the same database as the partition's source data.

Every cube has at least one partition, which contains the cube's data; a single partition is automatically created for a cube when the cube is defined. When you create a new partition for a cube, the new partition is added to the set of partitions that already exist for the cube. The cube reflects the combined data contained in all of its partitions. The division of a cube into partitions is not visible to end users of the cube. You can use the Partition Wizard to create or edit a partition.

Note You can create multiple partitions in a cube only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

Partitions are a powerful and flexible means of managing cubes, especially large cubes. For example, a cube containing sales information can contain a partition for the data of each past year and also partitions for each quarter of the current year. At the end of the year the four quarterly partitions can be merged into a single partition for the year.

Partitions can be stored using combinations of options for source data location, aggregation data location, storage mode, and aggregation design. This flexibility enables you to design cube storage strategies appropriate to your needs.

Data Sources and Storage

Each partition has a data source, which can be the same as or different from the data source of the partition's cube. If the same data source is used, the partition and the cube do not need to have the same fact table. If a different data source is used, it must reference a database that contains a set of tables that are essentially the same as those in the cube's schema. Some minor variations, such as the fact table name, are permitted.

Each partition can store its aggregate data on the Analysis server computer where the partition is defined; this is the default. Partition aggregate data can also be stored on another Analysis server computer; this partition is a remote partition.

Each partition has a storage mode, which determines whether the partition's aggregate data is stored on an Analysis server computer or in the database specified in the partition's data source. The storage mode also determines whether a copy of the partition's source data is stored on the Analysis server computer.

Each partition can have a different aggregation design, which determines the number and contents of the aggregations created for the partition. With the Storage Design Wizard, you can tailor a partition's aggregation design by specifying constraints for storage utilization or increase in query performance. With the Usage-Based Optimization Wizard, you can perform these same actions, and you can also optimize the aggregation design based on queries previously sent to the partition's cube. You can select the queries by which to optimize.

The aggregate data is designed by using the Storage Design Wizard and the Usage-Based Optimization Wizard, but it is created when the partition or its cube is processed.

Object Hierarchy

Partitions are immediately subordinate to the cube. The data of a cube is the combination of all of the data of the cube's partitions. If a partition is added or deleted, and the cube is then processed, the data of the cube changes. Changes to a partition and subsequent processing can also cause the data of its cube to change.

In the object hierarchy, the following objects are immediately subordinate to the partition:

- Data sources

  A partition has a single data source. By default, this is the data source of the partition's cube. A different data source can be selected from the data sources in the database or created during partition creation.

- Aggregations

  A partition's aggregations apply to only that partition.

Caution When a cube contains multiple partitions, if the partitions are defined or handled incorrectly, it is possible for the cube to contain incorrect data. It is essential that you understand the considerations that apply to multiple-partition cubes before you create one.

Working with Partitions

You can use the Partition Wizard to create or edit a partition.

In Analysis Manager a local partition is identified by the following icon.

A remote partition is identified by the following icon.

After partitions are created, aggregations are usually the next objects to be created.

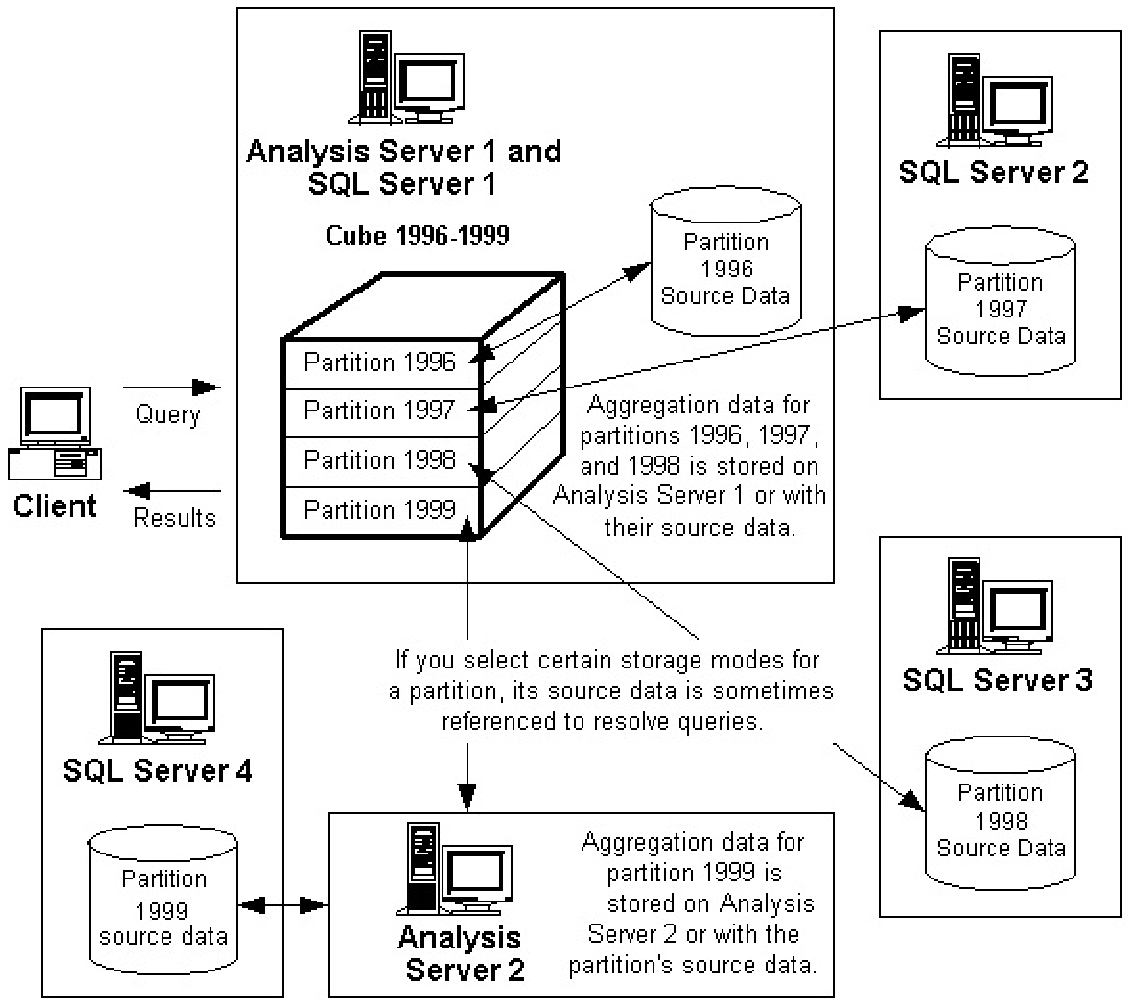
Structural changes to a cube, such as changes to its source data, fact table, or filter, require that you recreate the aggregations of all partitions in the cube.

You can merge two partitions that are in the same cube. The partitions to be merged must have the same storage mode and identical aggregation designs. The Partition Wizard provides options for copying the aggregation design of an existing partition.

If you are programming with Decision Support Objects (DSO), the class type associated with the partition is **clsPartition**.

Example: Four-Partition Cube

The following diagram shows how a four-partition cube can distribute its source data and aggregation data among five server computers. The activities of definition, storage design, processing, and querying are performed during four distinct time periods.



Analysis Server 1 and SQL Server 1
Cube 1996-1999
Partition 1996
Partition 1997
Partition 1998
Partition 1999
Partition 1996 Source Data
SQL Server 2
Partition 1997 Source Data
Client
Query
Results
Aggregation data for partitions 1996, 1997, and 1998 is stored on Analysis Server 1 or with their source data.
If you select certain storage modes for a partition, its source data is sometimes referenced to resolve queries.
SQL Server 3
Partition 1998 Source Data
SQL Server 4
Partition 1999 source data
Analysis Server 2
Aggregation data for partition 1999 is stored on Analysis Server 2 or with the partition's source data.

Cube 1996-1999 and its four partitions are defined on Analysis Server 1. The data sources of the partitions specify the locations of the source data of the partitions:

- The source data for Partition 1996 is stored on SQL Server 1, which is installed on the same computer as Analysis Server 1.
- The source data for Partition 1997 is stored on SQL Server 2.
- The source data for Partition 1998 is stored on SQL Server 3.
- The source data for Partition 1999 is stored on SQL Server 4.

Unlike partitions 1996, 1997, and 1998, Partition 1999 is defined as a remote partition. Its aggregate data is stored on a different Analysis server computer, Analysis Server 2. (Aggregate data is stored on the Analysis server computer only if the partition's storage mode is multidimensional OLAP (MOLAP) or hybrid OLAP (HOLAP); otherwise, data is stored in the same database as the partition's source data.)

Note Source data can be stored in databases of relational database products other than SQL Server. For more information, see Specifications and Limits.

Storage Design and Processing

For each partition in Cube 1996-1999, a storage mode is selected and aggregations are designed. The storage mode determines whether aggregate data will be stored on an Analysis server computer (that is, the storage mode is MOLAP or HOLAP) or with the source data of the partition (that is, the storage mode is relational OLAP (ROLAP)). The aggregation design determines the number of aggregations that are created.

When cube 1996-1999 is processed, aggregate data for its partitions is created. For partitions 1996, 1997, and 1998, aggregation data is stored on Analysis Server 1 or in the same databases as the source data of the partitions. For partition 1999, aggregation data is stored on Analysis Server 2 or in the same database as the partition's source data.

Queries

When a client that is connected to Analysis Server 1 issues a query to Cube 1996-1999, one or more of the following sources may be referenced to resolve the query:

- Client cache

  Whenever the value of a requested cell can be found in or derived from the results of a prior query that are stored in client cache, the value is retrieved from the client cache.

- Aggregation data

  If the value of a requested cell can be found in or derived from the aggregation data, the value is retrieved from the aggregation data.

- Source data

  If the value of a requested cell cannot be found in or derived from the aggregation data, the value is retrieved from the source data, provided that the partition has a storage mode of HOLAP or ROLAP.

- Copy of source data stored on the Analysis server computer

  If the value of a requested cell cannot be found in or derived from the aggregation data, the value is retrieved from a copy of the source data stored on the Analysis server computer, provided that the partition has a storage mode of MOLAP.

A single query can request data from multiple partitions.

See Also

Designing Storage Options and Aggregations

Managing Cube Storage

Optimizing Performance Based on Usage

Partition Wizard

Partition Structure

The structure of a partition must match the structure of its cube. The set of dimensions and measures that defines a cube's structure must also be present in the partition. For this reason, when a partition is created, it automatically inherits this set of measures and dimensions from its cube; it is not possible to edit this set within Analysis Manager.

However, it is possible for a partition and its cube to have different fact tables and/or data sources. Different partitions in a cube can have different fact tables and/or data sources. In these cases, a high degree of similarity among the fact tables and data sources of the partitions and the cube is required to ensure that the matching structures of the partitions and their cube are not undermined.

A cube and all of its partitions must have fact tables with the same structure. All columns used in the cube's measure definitions must be present and must be named the same in all of the fact tables. Extra columns can be present in some fact tables but absent in others. The fact tables can have different names.

All of the data sources of a cube and its partitions must reference databases that contain a schema identical to the schema in the definition of the cube. (You can display this schema in the Cube Editor **Schema** tab.) All dimension tables in the schema of the cube must be present and must be named the same in all of the databases. A fact table meeting the requirements described earlier in this topic must be present in each of the databases. All columns that join tables in the cube's schema must be present and must be named the same in all of the databases. The dimension tables must have the same structure in all of the databases. All columns used in the level definitions and member property definitions of the cubes must be present and must be named the same in all of the databases. Extra columns can be present in some databases but absent in others. Extra tables outside the required schema can be present in some databases but absent in others.

See Also

Cube Structure

Partition Storage

Physical storage options affect the performance, storage requirements, and storage locations of partitions and their parent cubes. One of these options is the storage mode of the partition. A partition can have one of three storage modes:

- Multidimensional OLAP (MOLAP)
- Relational OLAP (ROLAP)
- Hybrid OLAP (HOLAP)

Microsoft® SQL Server™ 2000 Analysis Services supports all three storage modes. With the Storage Design Wizard you can choose the storage mode most appropriate for your partition. Alternatively, you can use the Usage-Based Optimization Wizard to select a storage mode and optimize aggregation design based on queries that have been sent to the cube. Also, you can use an explicitly defined filter to restrict the source data that is read into the partition when using any of the three storage modes.

The MOLAP and ROLAP storage modes have somewhat different meanings when applied to dimensions and local cubes rather than partitions. The HOLAP storage mode does not apply to dimensions or local cubes.

MOLAP

The MOLAP storage mode causes the aggregations of the partition and a copy of its source data to be stored in a multidimensional structure on an Analysis server computer. This computer can be the Analysis server computer where the partition is defined or another Analysis server computer, depending on whether the partition is defined as local or remote. The multidimensional structure that stores the partition's data is located in a subfolder of the Data folder of the Analysis server. For more information about the Data folder, see Analysis Server.

Because a copy of the source data resides on the Analysis server computer, queries can be resolved without accessing the partition's source data even when the results cannot be obtained from the partition's aggregations. The MOLAP storage mode provides the potential for the most rapid query response times, depending on the percentage and design of the partition's aggregations. In general, MOLAP is more appropriate for partitions in cubes with frequent use and the necessity for rapid query response.

ROLAP

The ROLAP storage mode causes the aggregations of the partition to be stored in tables in the relational database specified in the partition's data source. However, you can use the ROLAP storage mode for the partition's data without creating aggregations in the relational database. For more information, see Set Aggregation Options (Storage Design Wizard) or Set Aggregation Options (Usage-Based Optimization Wizard).

Also, indexed views are created instead of tables if the partition's source data is stored in SQL Server 2000 and if certain criteria are met. For more information, see Indexed Views for ROLAP Partitions.

Unlike the MOLAP storage mode, ROLAP does not cause a copy of the source data to be stored; the partition's fact table is accessed to answer queries when the results cannot be derived from the aggregations or client cache. With the ROLAP storage mode, query response is generally slower than that available with the other two storage modes. ROLAP is typically used for large datasets that are infrequently queried, such as historical data from less recent previous years.

Note Aggregations cannot be created for a partition with ROLAP storage if the data source is Analysis Services (that is, if the provider is the Microsoft OLE DB Provider for Analysis Services).

HOLAP

The HOLAP storage mode combines attributes of both MOLAP and ROLAP. Like MOLAP, HOLAP causes the aggregations of the partition to be stored in a multidimensional structure on an Analysis server computer. HOLAP does not cause a copy of the source data to be stored. For queries that access only summary data contained in the aggregations of a partition, HOLAP is the equivalent of MOLAP. Queries that access source data, such as a drilldown to an atomic cube cell for which there is no aggregation data, must retrieve data from the relational database and will not be as fast as if the source data were stored in the MOLAP structure.

Partitions stored as HOLAP are smaller than equivalent MOLAP partitions and respond faster than ROLAP partitions for queries involving summary data. HOLAP storage mode is generally suitable for partitions in cubes that require rapid query response for summaries based on a large amount of source data.

See Also

Aggregations

Designing Storage Options and Aggregations

Optimizing Performance Based on Usage

Dimension Storage Modes

Indexed Views for ROLAP Partitions

If the storage mode of a partition is relational OLAP (ROLAP) and its source data is stored in Microsoft® SQL Server™ 2000, SQL Server 2000 Analysis Services attempts to create indexed views to contain aggregations of the partition. If Analysis Services cannot create indexed views, it automatically generates and uses aggregation tables instead of indexed views. While Analysis Services handles the session requirements for creating indexed views on SQL Server 2000, the creation and use of indexed views for aggregations requires the following conditions to be met by the ROLAP partition and the tables in its schema:

- The partition cannot contain measures that use the aggregate functions **Min** or **Max**.
- Each table in the schema of the ROLAP partition must be used only once. For example, the schema cannot contain "dbo"."address" AS "Customer Address" and "dbo"."address" AS "SalesRep Address".
- Each table must be a table, not a view.
- All table names in the partition's schema must be qualified with the owner name, for example, "dbo"."customer".
- All tables in the partition's schema must have the same owner; for example, you cannot have a FromClause like : "tk"."customer", "john"."store", or "dave"."sales_fact_1999".
- The source columns of the partition's measures must not be nullable.
- All tables used in the view must have been created with the following options set to ON:

    - ANSI_NULLS
    - QUOTED_IDENTIFIER
- The total size of the index key, in SQL Server 2000, cannot exceed 900 bytes. SQL Server 2000 will assert this condition based on the fixed length key columns when the CREATE INDEX statement is processed. However, if there are variable length columns in the index key, SQL Server 2000 will also assert this condition for every update to the base tables. Because different aggregations have different view definitions, ROLAP processing using indexed views can succeed or fail depending on the aggregation design.
- The session creating the indexed view must have the following options on: ARITHABORT, CONCAT_NULL_YEILDS_NULL, QUOTED_IDENTIFIER, ANSI_NULLS, ANSI_PADDING, and ANSI_WARNING. This setting can be made in SQL Server Enterprise Manager.
- The session creating the indexed view must have the following option off: NUMERIC_ROUNDABORT. This setting can be made in SQL Server Enterprise Manager.

For more information about the requirements of indexed views on SQL Server 2000, see Creating Indexed Views.

See Also

Real-Time Cubes

Remote Partitions

Remote Partitions

The data of a remote partition is stored on an Analysis server computer other than the one that contains the definitions (that is, meta data) of the partition and its parent cube. A remote partition is administered on the same Analysis server where the partition and its parent cube are defined.

When a remote partition is created, the Analysis server computer that stores its data is specified. This Analysis server computer is called the remote Analysis server of the remote partition.

Note The remote server computer must have Microsoft® SQL Server™ 2000 Analysis Services installed if the administering server computer runs SQL Server 2000 Analysis Services. Remote partitions on a server computer running SQL Server 7.0 OLAP Services are not supported.

Depending on the storage mode of a remote partition, the following data is stored on its remote Analysis server.

Storage type	Stores
MOLAP	The partition's aggregations and a copy of the partition's source data
HOLAP	The partition's aggregations
ROLAP	No partition data

Thus, if a cube contains multiple, remote multidimensional OLAP (MOLAP) or hybrid OLAP (HOLAP) partitions stored on multiple remote Analysis servers, the cube distributes its data among those Analysis servers.

When remote partitions are included in a cube, the memory and CPU utilization of a cube is also distributed. For example, when a remote partition is processed (alone or as part of its parent cube's processing), a majority of the memory and CPU utilization occurs on the remote Analysis server.

The meta data for a remote partition can be viewed and updated only on the Analysis server where the partition and its parent cube are defined. It cannot be viewed or updated on the remote Analysis server. Thus, a cube that contains remote partitions can be administered along with its partitions on a single Analysis server.

In Analysis Manager, a remote partition is identified by its unique icon.

To create a remote partition, use the Partition Wizard. In the **Specify the partition type** step, click **Remote**, and then select a remote Analysis server. For more information, see Creating Partitions.

To create or maintain a remote partition, your user name must be included in the OLAP Administrators group on both the Analysis server where the partition is defined and its remote Analysis server.

Remote partitions can be merged only with other remote partitions with the same remote Analysis server.

Data in remote partitions is not archived or restored with a database. After you restore a database that has a remote partition, you must process the remote partition. For more information about archiving and restoring databases, see Archiving and Restoring Databases. To process a partition, see How to process a partition.

If a remote partition is defined on an Analysis server, the logon account for the Analysis server service (MSSQLServerOLAPService) must be a domain user account. Otherwise, processing the remote partition will fail. If the account changes, you must either reprocess the remote partition or resave the partition. The remote partition can be resaved by either editing the partition in Analysis Manager using the Partition Wizard and finishing the wizard without changes, or by using a client application employing Decision Support Objects (DSO) to update the partition without changes. Either action will resynchronize the remote partition defined on the Analysis server.

Note Parallel processing of remote partitions is not supported.

A cube that contains a remote partition cannot contain a write-enabled dimension. For more information about write-enabled dimensions, see Write-Enabled Dimensions.

See Also

Partition Wizard

Incremental Updates and Partitions

When you create and manage partitions in multiple-partition cubes, you must take special precautions to ensure accurate cube data. Although these precautions do not usually apply to single-partition cubes, they do apply when you incrementally update them.

When you incrementally update any cube, a temporary partition is created and merged with an existing partition. When you run the Incremental Update Wizard, you specify the data source and fact table of the temporary partition. You can also specify a filter to limit the contents of the temporary partition. If the cube contains multiple partitions, you must specify the partition into which the temporary partition is merged. If the cube contains only one partition, the temporary partition is merged into that partition.

To ensure accurate cube data, before you perform an incremental update on any cube, be sure you understand the special precautions related to data integrity that apply to multiple-partition cubes. For more information, see Managing Partitions.

See Also

Processing Cubes

Analysis Services

Aggregations

Aggregations are precalculated summaries of data that improve query response time by having the answers ready before the questions are asked. For example, when a data warehouse fact table contains hundreds of thousands of rows, a query requesting the weekly sales totals for a particular product line can take a long time to answer if the fact table has to be scanned to compute the answer. However, the response can be almost immediate if the summarization data to answer this query has been precalculated. Precalculation of summary data is the foundation for the rapid response times of OLAP technology.

Cubes are the way OLAP technology organizes summary data into multidimensional structures. Dimensions and their hierarchical levels reflect the queries that can be asked of the cube. Aggregations are stored in the multidimensional structure in cells at coordinates specified by the dimensions. For example, the question "What were the sales of product X in 1998 for the Northwest region?" involves three dimensions (product, time, and geography) and one measure (sales). The value in the sales cell within the cube at the coordinates (product X, 1998, Northwest) is the answer, a single numerical value.

Other questions may return multiple values, such as "What were the sales of hardware products by quarter by region for 1998?" Such queries return sets of cells from the coordinates that satisfy the specified conditions. The number of cells returned by the query depends on the number of items in the hardware level of the product dimension, the four quarters in 1998, and the number of regions in the geography dimension. If all summary data has been precalculated into aggregations, the response time of queries like this will depend only on the time needed to extract the specified cells. No calculation or reading of data from the fact table is necessary.

Precalculation of all possible aggregations in a cube results in the fastest possible response time for all queries. However, the storage and processing time required for the aggregations can be substantial. Storage requirements depend not only on the number of dimensions and measures, but also on the number of levels in the dimensions and the number of members of each level.

There is a tradeoff between storage requirements and the percentage of possible aggregations that are precalculated. If no aggregations are precalculated (0%), little storage space is required beyond that necessary to store the base data. In this case, however, query response time will vary and may be quite slow because all answers will have to be calculated from the base data for each query. Returning the single number that answers the first query ("What were the sales of product X in 1998 for the Northwest region") might require reading thousands of rows of data, extracting the sale value from each, and calculating the sum.

Microsoft® SQL Server™ 2000 Analysis Services incorporates a sophisticated algorithm to select aggregations for precalculation so that other aggregations can be quickly computed from precalculated values. For example, if the aggregations are precalculated for the month level of a time dimension, the calculation for a quarter requires only the summarization of three numbers, which can be quickly computed on the fly. This technique saves storage with little effect on query response time.

The Storage Design Wizard provides options for you to specify storage and percentage constraints to the algorithm to achieve a satisfactory tradeoff between query response time and storage requirements. For more information about using the Storage Design Wizard, see Designing Storage Options and Aggregations. The Usage-Based Optimization Wizard enables you to adjust the aggregation design for a cube by analyzing the queries that have been submitted by client applications. You can tune a cube's aggregation design to provide rapid response to frequent queries and less rapid response to infrequent queries without substantially affecting the storage needed for the cube. For more information about using the Usage-Based Optimization Wizard, see Optimizing Performance Based on Usage.

Aggregations are designed in the preceding wizards but are created when the cube or partition for which the aggregations are designed is processed. After aggregation creation, if the structure of a cube changes, or if data is added to or changed in a cube's source tables, it is usually necessary to design the cube's aggregations again and process the cube again. For more information, see Processing Cubes.

In the object hierarchy shown in Object Architecture Overview, aggregations are immediately subordinate to the partition. Aggregations are always subordinate to a specific partition, but if a cube contains only one partition, its aggregations can be considered subordinate to the cube. For this reason, if the preceding wizards are run on a cube, selection of a partition is requested only if the cube contains multiple partitions.

Aggregations for a partition are stored in one of three locations, depending on the storage mode and type of the partition.

Storage mode	Type	Location of aggregations
Multidimensional OLAP (MOLAP) or hybrid OLAP (HOLAP)	Local	Subfolder of the Data folder of the Analysis server on which the partition is defined
Multidimensional OLAP (MOLAP) or hybrid OLAP (HOLAP)	Remote	Subfolder of the Data folder of the remote Analysis server specified when partition was created
Relational OLAP (ROLAP)	Local or remote	Dedicated tables or indexed views in the database specified in the data source of the partition

Note You can use the ROLAP storage mode for the data of a partition without creating aggregations in the relational database. For more information, see Set Aggregation Options (Storage Design Wizard) or Set Aggregation Options (Usage-Based Optimization Wizard).

For more information about the Data folder, see Analysis Server.

After aggregations are created, roles are usually the next objects to be created. For more information, see Roles.

If you are programming with Decision Support Objects (DSO), the class type associated with the aggregation is **clsAggregation**. For more information, see clsAggregation.

See Also

Partitions

Remote Partitions

Partition Storage

Analysis Services

Roles

Roles are used to control end users' access to cube data or data mining models while they are connected to the Analysis server with client applications. Each role definition includes a list of Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 user accounts and groups and specifies the types and scope of access they are allowed.

Microsoft SQL Server™ 2000 Analysis Services includes three types of roles:

- Database roles

  Defined at the Analysis Services database level, a database role can be assigned to multiple cubes in the database, thereby granting the role's users access to these cubes. Such an assignment creates a cube role with the same name as the database role. A database role provides defaults for cube roles of the same name. For more information, see Database Roles.

- Cube roles

  Created at the cube level when a database role is assigned to a cube, a cube role applies to only that cube. Defaults in a cube role are derived from the database role of the same name, but some of these defaults can be overridden in the cube role. A cube role contains additional options such as cell security that are not contained in a database role. For more information, see Cube Roles.

  You can exercise great flexibility in granting both read and read/write access to portions of cubes. You can specify which dimension members and cube cells a role can view and update. For more information, see Dimension Security and Cell Security.

- Mining model roles

  Similar to cube roles, mining model roles are created at the mining model level when a database role is assigned to a mining model; a mining model role applies to only that model. Defaults in a mining model role are derived from the database role of the same name, but role membership can be overridden in the mining model role. For more information, see Mining Model Roles.

See Also

Creating Security Roles

Database, Cube, and Mining Model Roles

Security and Authentication

Database Roles

A database role applies to a single database, and it includes a list of Windows NT® 4.0 or Windows® 2000 user accounts and groups. The specifications in a database role apply to end users' access to objects on an Analysis server while they are connected to it from a client application. Database roles are not used to grant or deny administrative access to objects.

Usually, database roles are created and then assigned to cubes or data mining models. Each assignment grants access to the cube or mining model and creates a cube or mining model role with the same name as the database role. The database role provides defaults for the cube or mining model role. The cube role can then be tailored to accommodate different permissions for various elements of the cube; however, customization of mining model roles is limited to changing role membership for access to the model.

In a database role, you can control access to the dimensions in the database. You can specify which levels and members of a dimension the role can view. If a dimension is write-enabled, you can specify which levels and members the role can update.

Database roles are immediately subordinate to the database. The database roles in a database apply to only that database and the objects within that database.

In the object hierarchy, commands are immediately subordinate to database roles. These commands are role commands. A role command is executed only for a specific database role. For example, a calculated member is created as a role command so that it is displayed only to end users in a particular database role. You must create and maintain role commands programmatically by using **clsRoleCommand**.

In Analysis Manager, database roles are identified by the following icon.

In Analysis Manager, database roles are created and maintained in Database Role Manager and the **Database Role** dialog box. Database roles are assigned to cubes in the **Cubes** tab of the **Database Role** dialog box or in Cube Role

Manager.

If you are programming with Decision Support Objects (DSO), the class type associated with the database role is **clsDatabaseRole**.

See Also

Administrator Security

clsDatabaseRole

clsRoleCommand

Creating Database Roles

Cube Roles

Database, Cube, and Mining Model Roles

Data Mining Models

Database Role Manager

Database Security

Dimension Security

Mining Model Roles

Cube Roles

A cube role applies to a single cube, includes a list of Windows NT® 4.0 or Windows® 2000 user accounts and groups, and indicates the objects in the cube they can access and the kind of access they have to those objects.

The specifications in a cube role apply to end users' access to a cube on an Analysis server while they are connected to it with a client application. Cube roles are not used to grant or deny administrative access to objects. For information about administrative access, see Administrator Security.

Usually, database roles are created and then assigned to cubes. Each assignment grants access to the cube and creates a cube role with the same name as the database role. The database role provides defaults for the cube role. The cube role can then be tailored for the cube.

In a cube role, you can control access to the dimensions in the cube. You can specify which levels and members of a dimension the role can view. If a dimension is write-enabled, you can specify which levels and members the role can update. For more information, see Dimension Security.

In a cube role, you can also control access to the cells in the cube. You can specify which cells the role can view. If a cube is write-enabled, you can specify which cells the role can update. For more information, see Cell Security.

In a cube role, you can also indicate whether the role can drill through to a cell's source data. To use this capability, you must also enable drillthrough for the cube or at least one of its partitions. For more information, see Specifying Drillthrough Options.

Cube roles are immediately subordinate to the cube. The cube roles in a cube apply to only that cube and the objects within that cube. However, changes to some specifications in a cube role propagate to the database role and all cube roles with the same name as the changed cube role. These specifications include the list of user accounts and groups and read/write permissions for dimensions.

In the object hierarchy, commands are immediately subordinate to cube roles. These commands are role commands. A role command is executed only for a specific cube role. For example, a calculated member is created as a role command so that it is displayed only to end users in a particular cube role. You must create and maintain role commands programmatically by using **clsRoleCommand**.

In Analysis Manager, cube roles are identified by the following icon.

In Analysis Manager, cube roles are created and maintained in Cube Role Manager and the **Cube Role** dialog box. Database roles are assigned to cubes in the **Cubes** tab of the **Database Role** dialog box or in Cube Role Manager.

Cube roles are usually the last objects created in order to deploy a cube.

If you are programming with Decision Support Objects (DSO), the class type associated with the cube role is **clsCubeRole**.

See Also

clsCubeRole

clsRoleCommand

Creating Cube Roles

Cube Role Manager

Cube Role Dialog Box

Cube Security

Database Roles

Database, Cube, and Mining Model Roles

Mining Model Roles

Object Architecture Overview

Mining Model Roles

A mining model role applies to a single mining model and includes a list of Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 user accounts and groups that have access to the model.

The specifications in a mining model role apply to end users' access to a mining model on an Analysis server while they are connected to it with a client application. Mining model roles are not used to grant or deny administrative access to models.

Usually, database roles are created and then assigned to mining models. Each assignment grants access to the model and creates a mining model role with the same name as the database role. The database role provides default membership for the mining model role, but users can be added or deleted from the default membership of the mining model role.

Mining model roles are immediately subordinate to the mining model. The mining model roles in a mining model apply to only that model. However, changes to role membership in a mining model role propagate to the database role and all mining model roles with the same name as the changed mining model role.

In Analysis Manager, mining model roles are created and maintained in Mining Model Role Manager and the **Mining Model Role** dialog box. Database roles are assigned to mining models in the **Mining Models** tab of the **Database Role** dialog box or in Mining Model Role Manager.

Mining model roles are usually the last objects created in order to deploy a mining model.

If you are programming with Decision Support Objects (DSO), the class type associated with the cube role is **clsMiningModelRole**.

See Also

Administrator Security

Data Mining Models

Mining Model Security

Analysis Services

Commands

A command is administrator-defined and is automatically executed when a client accesses a database, cube, or role. Although the term *command* is frequently used in a general sense, in this topic it refers to the command object, which is used in Analysis Services for specific purposes.

Commands are optional, but they are commonly used to enhance cubes.

You can create commands that enable end users to perform additional operations on cubes or to enrich or customize their experience in a client application. For example, you can create a type of command called an action, which allows end users to select a part of a cube and initiate an operation on it. For example, they can open their Internet browsers to display information about a particular member. Or you can create another type of command called a calculated member, which adds to a cube a customized member derived from an expression.

Commands can be defined for databases, cubes, database roles, and cube roles. Commands are immediately subordinate to these four objects. Commands defined for databases are called database commands. You must create and maintain database commands programmatically by using Decision Support Objects (DSO). Commands defined for cubes are called cube commands. You can create and maintain cube commands in Analysis Manager. You can also create and maintain cube commands programmatically by using DSO. Commands defined for database roles or cube roles are called role commands. They offer the flexibility of limiting commands to specific roles. You must create and maintain role commands programmatically by using the DSO.

A command is executed when an end user accesses a database, cube, database role, or cube role using a client application. Any of these actions trigger database commands defined for the database. Access to a cube occurs when any end user browses the cube or accesses its meta data. This action triggers various cube commands defined for the cube. Access to a database role occurs when an end user belonging to the database role accesses the database associated with the database role. This action triggers role commands defined for the database role. Access to a cube role occurs when an end user in the cube role browses the cube associated with the cube role. This action triggers role commands defined for the cube role.

You can create several types of commands, which are described in the following table. All of these types can be created as cube commands by using either Cube Editor for regular cubes or Virtual Cube Editor for virtual cubes.

Note Within a cube, commands such as actions, calculated members, or named sets cannot share the same name. Each command must have a unique name.

Command type	**Description**	**For more information, see**
Action	An end user-initiated operation upon a selected cube or portion of a cube. The operation can open an application with the selected item as a parameter or retrieve information about the selected item.	Actions
Calculated member	A dimension member whose value is calculated at run time using an expression that you specify when you define the calculated member.	Calculated Members
Named set	A set of dimension members or a set expression that is created for reuse, for example, in Multidimensional Expressions (MDX) queries.	Named Sets
Library registration	A registration of a file containing one or more functions for use in MDX.	Library Registrations

See Also

Decision Support Objects

Object Architecture Overview

Retrieving Schema Information

Actions

An action is an end user-initiated operation upon a selected cube or portion of a cube. The operation can start an application with the selected item as a parameter or retrieve information about the selected item.

Actions enable end users to act upon the outcomes of their analyses. By using actions, end users can go beyond traditional analysis and initiate solutions to discovered problems and deficiencies. Actions have the potential of transforming client applications from sophisticated data rendering tools to integral parts of the enterprise's operational system. Actions provide an easy way for end users to step from focusing on specific data to sending that data as input to operational applications.

For example, an end user browsing an Inventory cube notices that the current stock of white shoes is low. The end user selects the Order action on the white shoes member. This action initiates a new order for more white shoes in the order entry system.

You can exercise flexibility when you create actions. There are several types of actions for launching different kinds of applications and retrieving different kinds of information. Moreover, you can create actions that are triggered from various portions of cubes, including dimensions, levels, members, and cells. You can create multiple actions for the same portion of a cube. You can also pass string parameters to the launched applications and specify the action captions displayed to end users.

You can use the Action Wizard to create actions. For more information, see Creating Actions.

In order for an end user to use actions, his or her client application must support actions.

Although a command object is automatically executed when an end user accesses the object for which the command object is defined, if the command object is an action, the action itself is not automatically executed. The execution of the command creates the action and thereby makes it available to the end user. The action is executed when the end user performs the client-specific operation that initiates the action.

Actions created in the Action Wizard are saved so they can be used when browsing a cube that contains the actions. Actions can also be created that are in effect only for the duration of a session and are not saved. To create an action for the duration of a session, use the CREATE ACTION statement. This statement is also used in the **Statement** property when you create actions programmatically with Decision Support Objects (DSO). For more information, see CREATE ACTION Statement.

In Analysis Manager, an action is identified by the following icon.

Calculated Members

A calculated member is a dimension member whose value is calculated at run time using an expression that you specify when you define the calculated member. Calculated members can also be defined as measures. Only the definitions for calculated members are stored; values are calculated in memory when needed to answer a query.

Calculated members enable you to add members and measures to a cube without increasing its size. Although calculated members must be based on data (such as members) that already exists in the cube, you can create complex expressions by combining this data with arithmetic operators, numbers, and a variety of functions. Analysis Services includes a library of over 100 functions and allows you to register and use other function libraries.

For example, suppose executives in a shipping company want to determine which types of cargo are more profitable to carry, based on profit per unit of volume. They use a Shipments cube that contains the dimensions Cargo, Fleet, and Time and the measures Price_to_Ship, Cost_to_Ship, and Volume_in_Cubic_Meters; however, the cube does not contain a measure for profitability. You can create a calculated member as a measure named Profit_per_Cubic_Meter in the cube using the existing measures in the expression:

```
([Measures].[Price_to_Ship] - [Measures].[Cost_to_Ship]) /
[Measures].[Volume_in_Cubic_Meters]
```

The next time the Shipments cube is browsed, Profit_per_Cubic_Meter appears along with the other measures.

To create calculated members, use Calculated Member Builder. For more information, see Creating Calculated Members and Calculated Member Builder.

In Analysis Manager, a calculated member is identified by the following icon.

Calculated members have a **Format String** property that controls the format of cell values displayed to end users. This property is accessed in the properties pane of Cube Editor or Virtual Cube Editor. The **Format String** property accepts the same values as the **Display Format** property of measures.

Although calculated members created in Calculated Member Builder are saved so that they can be used when their cube is browsed, calculated members can also be created but not saved. To create a calculated member for the duration of a session, use the CREATE MEMBER statement. This statement is also used in the **Statement** property when you create calculated members programmatically with Decision Support Objects (DSO). To create a calculated member for the duration of a single query, use the WITH clause of the SELECT statement.

See Also

CREATE MEMBER Statement

Display Formats

Using WITH to Create Calculated Members

Named Sets

A named set is a set of dimension members or a set expression that is created to be used again. For example, a cube contains a Salesperson dimension. An end user of the cube wants an easy way to display the 10 salespersons with the highest sales and the 10 salespersons with the lowest sales. You can create two named sets, one for each group of salespersons. Then, in the client application, the end user can place the named sets on an axis in a manner similar to a dimension.

To create the named set for the 10 salespersons with the highest sales, use the following expression, written in Multidimensional Expressions (MDX), in the named set definition:

```
TopCount([Salesperson].[Salesperson Name].Members,10,[Measures].
```

To create the named set for the 10 salespersons with the lowest sales, use the following MDX expression in the named set definition:

```
BottomCount([Salesperson].[Salesperson Name].Members,10,[Measu
```

A named set can contain a constant set of members, or it can contain an expression that resolves to a set. If it contains an expression, as in the preceding examples, the set of members in the named set is subject to change.

To create named sets, use Named Set Builder.

In order for an end user to use named sets, his or her client application must support named sets.

Named sets created in the Named Set Builder are saved so they can be used when browsing a cube that contains the named sets. Named sets can also be created that are in effect only for the duration of a session and are not saved. To create a named set for the duration of a session, use the CREATE SET statement. This statement is also used in the **Statement** property when you create named sets programmatically with Decision Support Objects (DSO). To create a named set for the duration of a single query, use the WITH clause of the SELECT statement.

In Analysis Manager, a named set is identified by the following icon.

{..}

See Also

Creating Named Sets

CREATE SET Statement

Named Set Builder

Using WITH to Create Named Sets

Library Registrations

You can register files that contain functions to be used in Multidimensional Expressions (MDX). This is accomplished in Analysis Manager by using the **Register Function Libraries** dialog box. This dialog box is displayed by clicking **Register** in any of the following tools:

- MDX Builder
- Calculated Member Builder
- Named Set Builder

A command for a library registration includes the USE LIBRARY statement, which is created automatically if you register the library in the **Register Function Libraries** dialog box.

Note It is not necessary to register the Analysis Services function library, Microsoft® Excel worksheet function library, or Microsoft Visual Basic® for Applications Expression Services function library. These function libraries are registered automatically. However, the Microsoft Excel worksheet function library must be installed separately from Analysis Services to be registered.

See Also

Register Function Libraries Dialog Box

User-Defined Functions with MDX Syntax

Analysis Services

Member Properties

A member property is an attribute of a dimension member. It provides end users with additional information about the member. For more information about dimension members, see Dimension Structure.

The term member property sometimes refers to any of the properties of a member, including the standard set of properties associated with all members. However, in this topic, member property refers to the member property object in Microsoft® SQL Server™ 2000 Analysis Services. These objects are member properties you create in addition to the standard set.

Member properties are optional, but they are commonly used to enhance dimensions.

In some client applications, member properties can be easily viewed. The method of displaying member properties varies from application to application; for example, an end user might right-click a member to view its member properties. This presentation makes useful information available about members without cluttering the default view.

Member properties have a variety of uses. In addition to providing information about a member, member properties can be used in queries to provide end users with more options when analyzing cube data. Member properties can also be the basis of levels in virtual dimensions. For more information, see Virtual Dimensions.

Member properties are immediately subordinate to the level. A member property is created in the level that contains the members to which the member property applies.

You can create a member property in Dimension Editor by associating the column that contains values for the member property with the level that contains the members. For example, to create the member property Store Type for members of the Store Name level, ensure that the **Store Type** column is in the same table as the **Store Name** column, and then insert Store Type as a member property in the Store Name level.

You can create member properties in shared dimensions by using Dimension

Editor. In Analysis Manager, a member property is identified by the following icon.

If you are programming with Decision Support Objects (DSO), the class type associated with the member property is **clsMemberProperty**.

See Also

clsMemberProperty

Creating Member Properties

Dimension Editor - Data View

Multiple Language Implementation Using Member Properties

You can use the **Member Caption** property of a member property to display member captions and member property values in client locale-specific languages. For example, if an end user in France accesses a cube from a workstation with a French locale setting, the end user sees the member captions and member property values in French. However, if an end user in Germany accesses the same cube from a workstation with a German locale setting, the end user sees the captions names and member property values in German. Implementing multiple-language support for member captions is separate from yet similar to implementing multiple-language support for member property values. You can implement one without the other.

To implement multiple language support for member captions, the dimension table that contains the members must contain a column for each supported language. Each of these columns must contain the values for the member captions for one of the supported languages. It is best to dedicate one column for the default language. This column supplies member captions to end users when the locale settings of their workstations do not correspond to any of the specifically supported languages. When no member property of type **Caption** is defined, member names are used as member captions.

Note To implement multiple language support for member captions, the client application must use the MEMBER_CAPTION column of the members schema rowset and axis schema rowset to supply displayed member names.

To implement multiple language support for member property values, the dimension table that contains the members must contain a column for each supported language. The column must contain the member property values for one of the supported languages. It is best to dedicate one column for the default language. This column supplies member property values to end users when the locale settings of their workstations do not correspond to any of the specifically supported languages.

Background for Examples

An Inventory cube for a chain of coffee shops contains an Item dimension that contains members for the items the shops keep in supply. These members are in the Item Name level. A member property called Rotation Interval is planned for this level. After columns are added to support member captions and member property values in French, German, and English, the default language, the Item dimension table (named **Item**) looks like this.

Item ID	French Item Name	German Item Name	English Item Name	French Rotation Interval	German Rotation Interval	English Rotation Interval
1	Café	Kaffee	Coffee	Mensuellement	Monatlich	Monthly
2	Sucre	Zucker	Sugar	Annuellement	Jahrlich	Yearly
3	Crème	Kreme	Cream	Par semaine	Wochentlich	Weekly
4	Lait	Milch	Milk	Par semaine	Wochentlich	Weekly

Example for Member Names

To support member names displayed in French on French workstations, German on German workstations, and English on all other workstations, in the Item Name level, create the following three member properties with the following properties. You can set the properties in Dimension Editor.

- Member property for French
    - **Name** property value is **Member Name in French**
    - **Source Column** property value is **"Item"."French Item Name"**
    - **Language** property value is **French**
    - **Type** property value is **Caption**

    - **Caption** property value is **MEMBER_CAPTION**
- Member property for German
    - **Name** property value is **Member Name in German**
    - **Source Column** property value is **"Item"."German Item Name"**
    - **Language** property value is **German**
    - **Type** property value is **Caption**
    - **Caption** property value is **MEMBER_CAPTION**
- Member property for English, the default language
    - **Name** property value is **Member Name in English**
    - **Source Column** property value is **"Item"."English Item Name"**
    - **Language** property value is **(All)**
    - **Type** property value is **Caption**
    - **Caption** property value is **MEMBER_CAPTION**

The **Name** property values can vary.

The **Language** property value **(All)** for the English member property indicates that English is the default language.

These member properties do not appear as member properties in the client application.

Example for Member Property Values

To support Rotation Interval member property values displayed in French on French workstations, German on German workstations, and English on all other workstations, in the Item Name level, create the following three member properties with the following properties. You can set the properties in Dimension Editor.

- Member property for French
    - **Name** property value is **Rotation Interval in French**
    - **Source Column** property value is **"Item"."French Rotation Interval"**
    - **Language** property value is **French**
    - **Type** property value is **Regular**
    - **Caption** property value is **Rotation Interval**
- Member property for German
    - **Name** property value is **Rotation Interval in German**
    - **Source Column** property value is **"Item"."German Rotation Interval"**
    - **Language** property value is **German**
    - **Type** property value is **Regular**
    - **Caption** property value is **Rotation Interval**
- Member property for English, the default language

- **Name** property value is **Rotation Interval**
- **Source Column** property value is **"Item"."English Rotation Interval"**
- **Language** property value is **(All)**
- **Type** property value is **Regular**
- **Caption** property value is **Rotation Interval**

The **Name** property values can vary, but each should be in the user's language.

The **Language** property value **(All)** for the English member property indicates that English is the default language.

The **Caption** property values can be different from Rotation Interval but must be the same for all languages.

See Also

Dimension Editor - Data View

Analysis Services

Data Mining Models

In this release, Microsoft® SQL Server™ 2000 Analysis Services introduces data mining, a new feature that integrates significant data analysis and prediction capabilities into Analysis Services. In addition to extensions to Analysis Manager, extensions to Decision Support Objects (DSO) and PivotTable® Service have also been made to support this powerful new feature.

- Analysis Services supports data mining from both relational and multidimensional data sources. The Analysis Services algorithms can train data mining models with data from any relational data source that supports OLE DB access, as well as from multidimensional cubes created with Analysis Services.

- Extensibility allows Analysis Services to be used with third-party tools such as mining model viewer components, providing flexibility and enhancement.

- OLE DB support has been enhanced with the OLE DB for Data Mining specification.

- Analysis Manager has been upgraded with the addition of new wizards and tools to help design, create, train, and browse data mining models. Dimensions and virtual cubes can also be created based on OLAP data mining models.

- DSO has been extended with the addition of the **MiningModel** object and other ancillary support objects.

- Data mining is integrated into PivotTable Service following the same pattern that was used in Analysis Services. In general, a data mining model is treated like a cube by PivotTable Service: You can create local data mining models, retrieve information from server cubes, and so on.

Analysis Services

Introduction to Data Mining Models

A data mining model is the central object in data mining, one of the new features of Microsoft® SQL Server™ 2000 Analysis Services. A data mining model is a virtual structure that represents the grouping and predictive analysis of relational or multidimensional data. In many aspects, the structure of a data mining model resembles the structure of a database table. However, while a database table represents a collection of records, or a record set, a data mining model represents an interpretation of records as rules and patterns, composed of statistical information, referred to as cases. The structure of the data mining model represents the case set that defines the data mining model, while the data stored represents the rules and patterns learned from processing case data.

To understand what makes up cases and case sets, take for example a database designed to track customer orders. The database may contain a table for customer data, a table for order data, and a table for order items, shown here.

Each piece of information in a given table is a record. For each customer record, there may be one or more order records, each with one or more order item records. The relationship between order records and order item records implies that, for each customer, there may be many records in such a relationship. This collection of related records for a single customer is referred to as a case, and the same collection of related records for a group of customers is referred to as a case set. The order item information is treated as attributes of the customer case.

The case set is simply a way of viewing the physical data; in fact, different case sets can be constructed from the same physical data. The customer case set example is based upon the premise that you want to mine order item information with the customer as the focus. The focus could easily be changed to mining data about the customer with the order item as the focus. The physical data would not change, but a separate data mining model could easily be constructed to reflect the change in focus, with the customer information becoming attributes of the order item case.

Because of the innately hierarchical nature of such information, the data mining model stores the representation of a case set as a collection of data mining columns. Each data mining column can contain a group of data mining columns instead of a single data item such as a string or integer; each data mining column can contain single data items or another group of columns, and so on. In the customer case example, for each customer case, one row describes the customer. This row contains the customer ID and customer information columns, and a column named **Order Items**. The **Order Items** column contains a set of rows. Each row describes an order item that relates to the customer specified in the customer row. The following diagram illustrates the structure of such a case set.

In this example, some attributes of the customer, such as age and gender, might be used to further classify and predict the behavior of future customers. One of the most important tasks in data mining is to determine the impact of each of these attributes on classification and prediction.

Training a Data Mining Model

To determine the relative importance of each attribute in a data mining model, the model goes through a process known as mining model training. During training, data is supplied to the model for analysis. The data mining algorithms used by the model then examine the training data set in a variety of ways, to test it so that it can draw some conclusions about classification and prediction of the data.

For example, a decision tree mining model uses a process known as recursive partitioning to split the data up into partitions, based on the attributes supplied by the case set. Then, it splits up these newly created partitions into more partitions, and so on until no more useful splits can be performed. The algorithm itself determines what defines a useful split; this varies from technique to technique.

During this process of recursive partitioning, information is gathered from the attributes used to determine the split. If the **Age** column is used, for example, the model first divides the age values into two groups: those equal to or greater than a certain age, and those less than a certain age. By analyzing the number of records in the training set that fit one of the categories, a probability can be established for that category. As the splits grow, or increase in depth, more and more probability information can be gathered about the training data. When a decision tree can no longer split a given category usefully, that level of the tree is referred to as a leaf node. The leaf node contains information about the training data that fit that particular path through the decision tree. The information about the training in the leaf node is referred to as a distribution, and it is saved as part of the data mining model.

So, based on the training data set provided, the decision tree mining model establishes certain probabilities about the attributes in the customer case set. Applying those probabilities to other customer data, you can make predictions about customer behavior based upon the distribution information, or content, of the data mining model.

For more information about the data mining algorithms used, see Data Mining Algorithms.

Two objects are used to represent the structure of a case set in the Decision Support Objects (DSO) library. The **MiningModel** object holds the information about the data mining algorithms, queries and so on needed to describe and analyze the case set, as well as a collection of data mining column objects. In addition to containing information about its data type, each data mining column object holds attributes that describe its use within the data mining model, such as its relation to other data mining columns, whether it is used as a predictable column, whether it holds other columns, whether it is used as input for the data mining process, and so on. These data mining columns are represented by a collection of **Column** objects in the **MiningModel** object.

The data mining model is an abstract object; that is, the training data used to construct the case set is not saved. Rather, the abstraction of the model itself is saved, along with the results of the training data analysis, so that the same data mining model can be used with other data fitting its case set to provide predictive analysis.

For more information about the **MiningModel** object, see clsMiningModel.

For more information about the **Column** object, see clsColumn.

Integration with OLAP and Relational Data Sources

Data mining models can be trained using data from either an OLAP cube or a relational database. For relational databases, the only requirement is that the provider supports OLE DB. After a mining model has been created and trained, a connection to the original data source for the model is not required. For example, consider the following scenario:

> A large telephone company plans to roll out high-speed Internet access in a new market area. From experience in other market areas, the company has determined that persons who purchase high-speed Internet access fit a certain profile. The data that describes this profile is stored in a centrally managed relational database. A mining model is created that includes all of the elements (that is, characteristics) as columns. This model is then trained using the information from the previously existing market areas. This model can then be distributed to the new market areas for batch processing of the customers in that market. Additionally, the same model can be incorporated into the new service call center for the company, where the high-speed Internet service can be marketed to new customers that match that specific profile. In either situation, the original data from the previously existing markets is not needed to make a prediction of the Internet needs of the customer. The model contains within itself all of the information that is needed to make a prediction.

Data Mining Model Structure

The structure of a data mining model is defined primarily by a set of data mining columns and a data mining algorithm. The data mining model content, created by the training process, is stored as data mining model nodes.

Each data mining column can contain one of several different content types, depending upon its use within the data mining model. Each column type has its own properties and behaviors. For more information, see Data Mining Columns.

The data mining algorithm uses the data mining column definitions to generate a predictive model by running the algorithm on training data submitted to the data mining model. The data mining model then stores the results obtained from analyzing the training data. Even though large amounts of training data may be inserted into a data mining model, the training data itself is not stored. Only the analysis information gained by processing that data and the distinct column values used as part of the analysis are stored as data mining model content.

Data Mining Model Nodes

Data mining model nodes represent the content of a data mining model. Each node contains information about the attributes needed to define the node, the relevant rules and other information needed to process a case against the node, and the analysis gained from training the node. Each node can also be related to other nodes, to support the complexities of decision tree and clustering algorithms in a common structure. The data mining model nodes can be browsed to further understand the decisions or aggregations made by the algorithm employed, and they can be modified to further adjust the data mining model.

See Also

Data Mining Columns

Data Mining Algorithms

Data Mining Model Nodes

Data Mining Algorithms

Central to the data mining process, data mining algorithms determine how the cases for a data mining model are analyzed. Data mining model algorithms provide the decision-making capabilities needed to classify, segment, associate and analyze data for the processing of data mining columns that provide predictive, variance, or probability information about the case set.

Many data mining algorithms are goal-oriented; given a case set, a data mining algorithm will predict something about the case, usually an attribute of the case itself. Most algorithms require a training set of cases where the attributes to be predicted are already known, at which point the algorithm constructs a data mining model capable of predicting these attributes for cases in which the attributes are unknown. For more information about training data mining models, see Introduction to Data Mining Models.

Each data mining algorithm is supported by a data mining algorithm provider, which is an OLE DB provider that supports the OLE DB for Data Mining specification. Because the needs and functions of each data mining algorithm provider are different, it may be necessary for a client application to first determine the capabilities of a data mining algorithm provider.

Not all data mining algorithm providers support all data mining options. Some providers may work with certain data mining column data or content types, and other providers may not support certain options for source data queries. To determine the capabilities of a data mining algorithm provider, the MINING_SERVICES schema rowset details data mining support options for each provider. Also, as each provider is an OLE DB provider, the standard OLE DB provider schema rowsets, such as the PROVIDER_TYPES schema rowset, can be used to give additional information.

Data Mining Algorithm Providers

Data mining algorithms fall into three general categories. This is not a comprehensive list of the various data mining algorithms that might be used; other data mining algorithm providers may be constructed based on, for example, back propagation neural network or genetic algorithms.

Decision Trees

A decision tree is a form of classification shown in a tree structure, in which a node in the tree structure represents each question used to further classify data. The various methods used to create decision trees have been used widely for decades, and there is a large body of work describing these statistical techniques. For more information about the decision trees technique and the Microsoft® Decision Trees algorithm, see Microsoft Decision Trees.

Clustering

Like decision trees, clustering is a well-documented data mining technique. Clustering is the classification of data into groups based on specific criteria. The topic discussing the Microsoft Clustering algorithm goes into greater detail regarding the details of clustering as a data mining technique. For more information about the clustering technique and the Microsoft Clustering algorithm, see Microsoft Clustering.

Microsoft Decision Trees

The Microsoft® Decision Trees algorithm is based upon the notion of classification. The algorithm builds a tree that will predict the value of a column based upon the remaining columns in the training set. Therefore, each node in the tree represents a particular case for a column. The decision on where to place this node is made by the algorithm, and a node at a different depth than its siblings may represent different cases of each column. For instance, consider the following training table.

Shares files	Uses scanner	Infected before	Risk
Yes	Yes	No	High
Yes	No	No	High
No	No	Yes	Medium
Yes	Yes	Yes	Low
Yes	Yes	No	High
No	Yes	No	Low
Yes	No	Yes	High

For this training data, the following decision tree may be produced.

Notice that for users that share files, the most important factor (that is to say, training column) for determining their risk of computer virus infect is **Infected Before**. For users who don't share files, the most important factor is **Uses Scanner**. This demonstrates on of the key concepts behind the decision tree algorithm: A column may be used at more than one location in the tree, and its importance in the prediction may therefore change.

Mining Parameters

The Microsoft Decision Trees algorithm provider currently supports two mining parameters, which can be used to change the behavior of the algorithm when creating a model with the CREATE MINING MODEL command. The parameters are defined in the MINING_PARAMETERS schema rowset; a description of each parameter is provided in the following table.

Parameter	Description
COMPLEXITY_PENALTY	A floating point number with a range between 0 and 1. Used to inhibit the growth of the decision tree, the value is subtracted from 1 and used as a factor in determining the likelihood of a split. The deeper the branch of a decision tree, the less likely a split becomes; the complexity penalty influences that likelihood. A low complexity penalty increases the likelihood of a split, while a high complexity penalty decreases the likelihood of a split. The effect of this mining parameter is dependent on the mining model itself; some experimentation and observation may be required to accurately tune the data mining model. The default value is based on the number of attributes for a given model: • For 1 to 9 attributes, the value is 0.5. • For 10 to 99 attributes, the value is 0.9. • For 100 or more attributes, the value is 0.99.
MINIMUM_LEAF_CASES	A non-negative integer with a range of 0 to 2,147,483,647. Determines the minimum number of leaf cases required to generate a split in the decision tree. A low value causes more splits in the decision tree, but can increase the likelihood of overfitting. A high value reduces the number of splits in the decision tree, but can inhibit the growth of the decision tree. The default value is 10.

See Also

MINING_SERVICE_PARAMETERS

CREATE MINING MODEL Statement

Microsoft Clustering

The Microsoft® Clustering algorithm is an expectation method that uses iterative refinement techniques to group records into neighborhoods (clusters) that exhibit similar, predictable characteristics. Often, these characteristics may be hidden or nonintuitive. For example, suppose that a travel firm wants to determine age demographics for marketing vacation packages. From their data warehouse they have the following training data.

Customer age	Country traveled to
23	Mexico
45	Canada
32	Canada
47	Canada
46	Canada
34	Canada
51	Canada
28	Mexico
49	Canada
29	Mexico
26	Mexico
31	Canada

When this information is plotted on a graph with two dimensions, you can see that there are three main groups in the data: People between the ages of 23 and 29 seem to travel to Mexico. People between the ages of 30 and 51 seem to travel to Canada. The clustering algorithm also presents an interesting fact that might not be apparent from observing the data directly: People between the ages of 35 and 44 did not seem to travel at all. Another way of saying this is that the grouping of people who travel to Canada falls into two main clusters: People between the ages of 30 and 34, and people between the ages of 45 and 51.

The clusters of data in this example are readily observed. For data with higher dimensions, plotting the data in this manner may not be convenient, or the dimensions may not be amenable to plotting at all. The clustering algorithms automatically find such groupings in data with higher numbers of dimensions.

Mining Parameters

The Microsoft Clustering algorithm provider does not currently support any additional mining parameters.

Data Mining Model Nodes

When a data mining model is built and trained, the resulting data mining model content is stored as data mining model nodes. A node stores the attributes, description, probabilities, and distribution information for the model element it represents, as well as any cardinality information the node may possess in relation to other nodes.

Node Types

Each node has an associated node type that aids in representing a data mining model. The node types are used primarily for navigation, not as a way of defining functionality for the node. For example, although each node of a decision tree model may have a distribution associated with it, not all nodes in a decision tree model will be classified as distribution nodes. There are six types of currently supported nodes.

Model

A model node is the topmost node in any data mining model, regardless of the actual structure of the model. All models start with a model node.

Tree

For all tree-based models, this node serves as the root node of the tree. A data mining model may have many trees that make up the whole, but there is only one tree node from which all other nodes are related for each tree. A decision tree based model always has one model node and at least one tree node.

Interior

An interior node represents a generic interior node of a model. For example, in a decision tree, this node usually represents a split in the tree.

Distribution

A distribution node is guaranteed to have a valid link to a nested distribution table. A distribution node describes the distribution of values for one or more attributes according to the data represented by this node. A good example of a distribution node is the leaf node of a decision tree.

Cluster

A cluster node stores the attributes and data for the abstraction of a specific cluster. In other words, it stores the set of distributions that constitute a cluster of cases for the data mining model. A clustering based model always has one model node and at least one cluster node.

Unknown

The unknown node type is used when a node does not fit any of the other node types provided and the algorithm cannot resolve the node type.

The following diagram illustrates the differences between various node types and the algorithms that support them.

Browsing Data Mining Model Nodes

The nodes of a trained data mining model can provide valuable insight into the data. Nodes define the patterns and rules created by the analysis of the training data, and they can provide more information about new data predictions. The ability to browse a data mining model allows for refinement of the model with fine detail. Depending on the specific data mining algorithm used in the creation of the data mining model, the content type may vary on a model by model basis.

Data mining model content can be browsed in several ways.

Analysis Services

Analysis Manager provides Data Mining Model Browser, a useful tool for graphically exploring a data mining model and its content. For more information, see Data Mining Model Browser.

Rowset

Querying the model directly returns the data mining model content in the form of a single rowset.

SELECT * FROM <*mining model*>.CONTENT

The attributes and results for the nodes are stored in MINING_MODEL_CONTENT, a special schema rowset which allows for browsing of the data mining model content.

For more information about the storage schema, see Data Mining Model Storage.

XML

Another way to browse the content of a data mining model is as an Extensible Markup Language (XML) document. The XML information, however, is best viewed by a client application capable of parsing this complex data.

For more information about the document type definition (DTD) of the XML document, see the OLE DB for Data Mining specification.

Data Mining Model Storage

The data mining model meta data can be obtained by a client application using the MINING_MODELS schema rowset, while the schema of the data mining columns for a data mining model is stored in the MINING_COLUMNS schema rowset.

Content for the data mining model is stored in the MINING_MODEL_CONTENT schema rowset. Statistical distribution information for the attributes corresponding to a data mining model node stored in MINING_MODEL_CONTENT is represented as a nested table in the DISTRIBUTION column (represented in OLE DB as a chapter column) using the DISTRIBUTION structure.

See Also

MINING_MODELS

MINING_MODEL_CONTENT

MINING_COLUMNS

Distribution (clsColumn)

Analysis Services

Data Mining Columns

Data mining columns are used to define the inputs and outputs used by a data mining model. The data mining column also provides a standard structure against which familiar SQL syntax, such as INSERT for training data and SELECT for predictive analysis, can be used.

The structure and behavior of data mining columns can be viewed and changed by using Relational Mining Model Editor or OLAP Mining Model Editor. In both editors, the structure pane contains the data mining columns used to define the data mining model; the properties for each data mining column, such as data type and content type, can be viewed in the properties pane.

Data mining columns are added to the data mining model at different steps in the Mining Model Wizard, depending on the type of data mining model. For relational data mining models, the **Select the key column** and **Select input and predictable columns** steps add the key, input, and predictable data mining columns to the data mining model. For OLAP data mining models, however, three steps are used. The **Select case** step selects the case dimension and level used to create key data mining columns, the **Select the predicted entity** step creates the predictable data mining columns, and the **Select training data** step creates the input data mining columns.

Data Mining Column Structure

A data mining column is defined primarily by its data type and content type settings. These settings are detailed in other topics. Because of the diversity of possible data mining algorithm providers, the data mining column definitions are designed to be flexible and extensible.

See Also

Relational Mining Model Editor

OLAP Mining Model Editor

Data Mining Models

Data Mining Column Data Types

Data Mining Column Content Types

clsMiningModel

clsColumn

Data Mining Column Data Types

The data mining column data type is used as a tool to help the data mining algorithm provider handle input from training data and format output data for analysis.

You should not confuse the column data type with the column content type. For example, a column with a data type of **TEXT** can be used as a key, attribute, or relation column with equal ease. The column data type serves the data mining algorithm provider as a guideline for converting and processing the training data for the data mining column.

Supported data types are listed in Appendix A of the OLE DB 2.6 specification of March 1999 (version 2.6).

Note Not all data mining algorithm providers support all data types. For more information about supported column data types, see the data mining algorithm provider documentation.

Data Mining Column Content Types

There are five basic column content types, each of which is described later in this topic. Do not confuse column content type with column data type; a column content type provides the role that a column fulfills for a data mining model. A key column, for example, can have a column data type of **LONG** or **TEXT** and still serve the role of key column. Attribute and table columns, two column types discussed later in this topic, can also be used as prediction columns.

The column types and their associated properties allow the data mining algorithm provider to make some sense of the training data provided to it during the training process.

The following case diagram is used later in the topic to explain column types.

For more information about cases, see Introduction to Data Mining Models.

Key Columns

A key column (or columns) uniquely identifies a row. For example, the **CustomerID** and **OrderItemID** columns in the case diagram represent key columns. A case may be uniquely identified by one or more key columns.

Attribute Columns

Attribute columns provide information about direct attributes of the case. In the case diagram, the **Age** and **Gender** columns both represent attribute columns. An attribute column is further defined by a domain and handling hints.

Domains

A domain, or the set of possible values that can appear in the attribute column, further defines attribute columns. Domains are classified into a few simple groups, detailed in the following list.

DISCRETE

The values for the attribute column are discrete; this is the simplest form of attribute column. The **Gender** column in the case diagram represents a typical discrete attribute column, in that the data represents a finite, counted number of gender categories. The values in a discrete attribute column do not imply ordered data, even if numeric; the values are clearly separated, with no possibility of fractional values. Telephone area codes are a good example of numeric discrete data.

ORDERED

The values for the attribute column define an ordered set. Although there is an ordered set, no distance or magnitude information is implied. For example, if an ordered attribute column supplying information about a ranking of skill levels ranging from one to five is defined, there is no relative value between skill levels; a skill level of five is not necessarily five times better than a skill level of one. Ordered attribute columns are also considered to be discrete in terms of content type.

CYCLICAL

The values for the attribute column define a cyclical ordered set. An example of a cyclical ordered set is the numbered days of the week, as day number one follows day number seven.

Cyclical attribute columns are considered both ordered and discrete in terms of content type.

CONTINUOUS

The values for the attribute column define a continuous curve set. The values in the continuous curve are naturally ordered and have implicit distance and magnitude semantics. Unlike a discrete column, which represents finite, counted data, a continuous column represents measurement data with possibly infinite fractions of values. The **Age** column in the case diagram is an example of a continuous attribute column; the values in this column represent a measurement, in years, that could be represented by an infinite number of fractional values, such as 21.1, 21.003, and so on.

DISCRETIZED

The values for the attribute column define an ordered set transformed and modeled from continuous data supplied to the model. Some data mining algorithm providers cannot accept continuous attribute columns as input, or they may not be able to predict continuous values. For these cases, columns that have continuous domains can be employed as discretized attribute columns, in which the continuous values are grouped into discrete categories, so that the continuous data can be treated as discrete data for the purpose of analysis. For some data mining algorithm providers, discretized columns can take arguments to override default discretization behavior. The following list details currently supported discretization behavior flags.

AUTOMATIC

The data mining algorithm provider selects its default discretization method.

EQUAL_AREAS

The data mining algorithm provider attempts to divide the data into groups containing an equal number of continuous values. This method is best used for normal distribution curves, but will not work well for a high count of values in a narrow group in the continuous data. For example, if half of the order items specified in the case diagram are free, or have a Cost value of zero, then half the data is under a single point in the curve. For such a distribution, this method attempts to break such data up as part of establishing equal area discretization into multiple areas, producing undesirable results.

THRESHOLDS

The data mining algorithm provider attempts to divide the data into groups by reviewing the curve of the continuous data and searching for inflection points; this works well for continuous data that does not conform to a normal distribution, as the inflection points of such data can suggest reasonable boundaries for discretization.

CLUSTERS

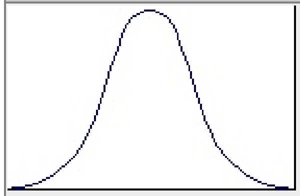
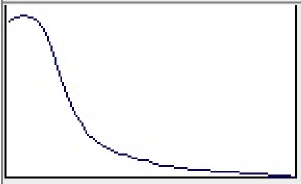
The data mining algorithm provider attempts to divide the data into groups by sampling the training data, initializing to a number of random points, and running several iterations of the Expectation-Maximization (EM) clustering algorithm. This method is beneficial in that it will work on any distribution curve, but is more expensive in terms of processing time.

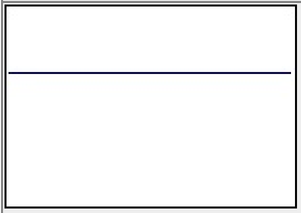
SEQUENCE_TIME

The values for the attribute column represent time measurement units. A time column does not have to contain a data type of any particular format; for example, a number representing periods or quarters is acceptable. The sequence time attribute column is typically used to associate a sequence time with individual attribute values such as purchase time.

Distributions

The domain of an attribute column classified as Continuous can also have a distribution associated with it. This is information given to the data mining algorithm provider describing the expected distribution of the column values that will be inserted into the model when trained. Specific values may be known to have typical distributions. For some algorithms, it is particularly beneficial to know the distribution ahead of time. If the distribution is not known or is not given, the provider may assume whatever distribution it finds convenient. Examples and diagrams of some distribution flags are included in the following table.

Diagram	Distribution
	**NORMAL** The values for the continuous attribute column form a histogram with a normal Gaussian distribution. For example, income values may form such a distribution curve.
	**LOG_NORMAL** The values for the continuous attribute column form a Gaussian distribution histogram with all values greater than zero, where the curve possesses an elongated upper tail and a skew toward the low end of the curve. The quantity associated with the number of order items purchased would follow this curve if a value of zero is not explicitly recorded and most consumers tend to buy smaller numbers of the

	order item.
	**UNIFORM** The values for the continuous attribute column form a flat curve, in which all values are equally likely.

Other supported distribution flags include:

- **BINOMIAL**
- **MULTINOMIAL**
- **POISSON**
- **T-DISTRIBUTION**

Hints

Other information can be given to the data mining algorithm provider to help it build good models of the training data. These modeling flags are provider-specific, but supported examples include the following.

MODEL_EXISTENCE_ONLY

The values for the attribute column are less important than the presence of the attribute. For example, the case diagram displays a list of order items associated with a given customer, including the ID, cost and product type of the order item. For modeling purposes, the fact that the customer purchased a given order item may be more important than the cost of the order item itself. In this case, the **Cost** column should be marked as Model Existence Only.

NOT NULL

The values for the attribute column should never contain a null value, and an error will result if a null value is encountered for this attribute column during the training process.

IGNORE NULL

The values for the attribute column can contain a null value, and the null value should not be considered informative by the data mining model. Null values will be ignored if encountered for this attribute column during the training process.

NULL INFORMATIVE

The values for the attribute column can contain a null value, and the null value should be considered informative by the data mining model. Null values will be modeled as a missing state if encountered for this attribute column during the training process.

Qualifier Columns

A qualifier column is a special type of attribute column that provides information about another attribute column to the data mining algorithm provider. The following list details currently supported qualifier column types, but third-party providers can add additional qualifier column types.

PROBABILITY

The value in this attribute column is the probability, as a number between zero and one, of the associated value.

VARIANCE

The value in this attribute column is the variance of the associated value.

STDEV

The value in this attribute column is the standard deviation of the associated value.

SUPPORT

The value in this attribute column is the weight (case replication factor) of the associated value.

PROBABILITY_VARIANCE

The value in this attribute column is the variance of the probability for the associated value.

PROBABILITY_STDEV

The value in this attribute column is the standard deviation of the probability for the associated value.

ORDER

The value in this attribute column is the ordering of the associated value.

Relation Columns

A relation column is a column used in a case to further classify another attribute, relation or key column, by pointing out a hierarchical relationship between attribute columns within the case set. For example, the **ProductType** column further classifies the **ProductID** column in the case diagram presented earlier in this topic; it provides a classification into which certain products fit. In effect, it establishes a hierarchy in that all products belong to a product type. A given relation value must always be consistent for all of the instance values of the other column that it describes. If a product is associated with a given product type, it must always be associated in the data with that product type.

Table Columns

A table column is a column that represents a set of data mining columns, also known as nested columns; it is described by a set of columns that are contained within the definition of a named table column. The **OrderItems** column in the case diagram represents a table column.

Prediction Columns

Attribute or table columns can be used as input columns, output columns, or both. The data mining algorithm provider will build a data mining model capable of predicting or explaining output column values based on the values of the input columns.

Prediction columns, besides serving as output columns, can also be used as input columns for other prediction columns within a case, allowing for complex predictive analysis. Not all data mining algorithm providers support all content types for prediction purposes; the capabilities of each data mining algorithm provider can be checked in the MINING_SERVICES schema rowset.

For more information about the MINING_SERVICES schema rowset, see MINING_SERVICES.

Predictions can convey not only simple information like the estimated age of a customer, but they can also convey additional statistical information, such as the confidence level and standard deviation. Further, the prediction may actually be a collection of predictions, such as the set of order items a customer is likely to buy based on a specific customer case. Each of the predictions in the collection may also include a set of statistics accompanying it, expressed as a histogram. A histogram provides multiple possible prediction values, each accompanied by a probability and other statistics. In this case, each prediction (which by itself can be part of a collection of predictions) may have a collection of possible values that constitutes a histogram.

Because the prediction information can be very detailed and complex, it is often necessary to extract only a portion of the prediction. For example, you may want to examine a specific prediction value or a range of values. Not every provider and every data mining model can support all of the possible requests. Therefore, it is necessary for the output column to indicate what kind of information may be extracted out of it, using transformation functions.

A set of standard transformation functions has been defined as part of the OLE DB for Data Mining specification.

For more information about the OLE DB for Data Mining specification, see the Microsoft OLE DB Web page at the Microsoft Web site.

Nested Data Mining Columns

One of the more advanced features of the data mining column structure is the ability to nest data mining columns. Data mining models use this nested column structure for both input and output data, as the syntax used to populate a data mining model with training data allows nested columns to be represented as subqueries. Data mining cases may not be easily described by using typical relational tables; a single case may depend on several groups of supporting information to perform predictive analysis. To illustrate this point, consider the case of a telephone company customer: A customer may have multiple telephone lines and multiple ISP accounts.

To retrieve all of the customer information, all of the telephone lines for each customer, and all of the ISP accounts for each customer, several approaches could be used:

- Employ three queries, iterating through the customer query and issuing the same two queries for retrieving telephone lines and ISP accounts over and over again for each row in the customer query.

- Employ two queries, joining the **Customers** table with the **Telephone Lines** table for one query and joining the **Customers** table with the **ISP Accounts** table for the second query.

- Employ one query with a UNION to join the two subqueries used in the previous approach, constructing empty columns in each subquery to represent the return values of the other subquery.

All three of these approaches are ungainly, involve repetitive data and action, and are highly inefficient.

However, if a single column could hold a group of columns, you could construct a single query that would return one row per customer in the **Customers** table containing all of the columns in the **Customers** table, an additional column containing all of the **Telephone Lines** rows for a given customer, and an additional column containing all of the **ISP Accounts** rows for a given customer, as shown in the following diagram.

As the diagram shows, there is no redundant data for the customer in the returned rowset; one row per customer is all that is needed, and the nested columns of the rowset contain the data pertinent to that customer. Rowsets constructed in this fashion, referred to as hierarchical rowsets, are fully supported by OLE DB.

Case information for a data mining model may not reside in a single case table, but may have supporting tables supplying additional information to define the case. In the diagram, the **Telephone Lines** and **ISP Accounts** tables serve as supporting tables for the **Customers** case table. They provide additional information about the case, such as the number and type of ISP accounts the customer may possess, or the number of telephone lines used by the customer. The data mining model can take advantage of nested data mining columns to process this supporting information and create additional rules and patterns for the customer based on the data in the supporting tables.

Data Mining Column Storage

The schema information for data mining columns is accessed by client applications by the use of the MINING_COLUMNS schema rowset, which contains not only the data and content representation of the column, but also information on its relationships within the data mining model, the scalar and table functions that the column supports, and so on. For more information, see MINING_COLUMNS.

Analysis Services

Security and Authentication

You can restrict access to data managed by Microsoft® SQL Server™ 2000 Analysis Services. You can limit the administrators that are permitted to access Analysis Services data through Analysis Manager and perform administrative functions. You can also restrict end users who access data on the Analysis server through client applications. You can specify which end users can access data and the types of operations they can perform. In addition, you can control end-user access at various levels of Analysis Services data, including the cube, dimension, and cube cell.

Administrator security is controlled using the Microsoft Windows NT® 4.0 or Windows® 2000 group named OLAP Administrators.

End-user security is controlled using:

- Authentication during connection to the Analysis server.
- Database, cube, and mining model roles defined in Analysis Manager.

Each role defines a set of users and the access they all share. A role is defined at the Analysis Services database level and then assigned to cubes that the users in the role are permitted to access. After assignment some changes are permitted to the role at the cube level. These changes do not affect the role at the database level. (An exception is a change to the users and groups in the cube role.) Thus, a role can have a different definition for each cube to which it is assigned.

Analysis Services supports Windows integrated security system.

The following topics describe security in more detail.

Topic	Description
Administrator Security	Provides information on administrative security roles
End-User Security	Identifies issues related to implementing end-user security
User Accounts and Groups	Provides information about establishing user

	accounts necessary for Analysis Manager roles
Database, Cube, and Mining Model Roles	Describes the three types of security roles used in Analysis Services
Levels of End-User Security	Identifies the Analysis Services objects that security roles can be applied to
Server Security and Authentication	Provides information about how server security is applied in Analysis Services using authentication
Database Security	Describes how database security roles are used in relation to database security
Cube Security	Describes how cube security roles are used in relation to cube security
Dimension Security	Describes how dimension security specifications are used in relation to cube security
Cell Security	Describes how cell security specifications are used in relation to cube security
Operational Considerations	Provides information about Analysis server procedures that should be followed to ensure system security
Mining Model Security	Provides information about how mining model roles are used for mining model security

Analysis Services

Administrator Security

Administrator security concerns users' access to Microsoft® SQL Server™ 2000 Analysis Services data through Analysis Manager and their ability to perform administrative functions. Administrator security is granted through membership in the OLAP Administrators group.

By default, the user account used to install the Analysis server on a particular computer is granted administrative permissions on that computer. Upon initial installation, Analysis Services establishes a Microsoft Windows NT® 4.0 or Windows® 2000 group named OLAP Administrators and adds the logged-on user to this group. The OLAP Administrators group is a local group on the computer where Analysis Services is installed. Only members of this group can:

- Access the Analysis server through Analysis Manager and perform administrative functions.

- Perform administrative functions on the Analysis server programmatically with Decision Support Objects (DSO).

Administrative functions include maintaining security roles and processing Analysis Services objects.

You can use User Manager in Windows NT 4.0 or Computer Management in Windows 2000 to manage the OLAP Administrators group.

There are not multiple degrees or levels of administrator security. A user either is or is not an administrator, depending on whether he or she is included in the OLAP Administrators group.

When connected to an Analysis server through client applications, members of the OLAP Administrators group have full read access to all cubes and dimensions on the server. They also have full read/write access to all write-enabled cubes and write-enabled dimensions. This access is granted regardless of role definitions on the Analysis server.

See Also

Operational Considerations

Analysis Services

End-User Security

End-user security is concerned with users' access to data on the Analysis server through client applications such as Microsoft® Excel. It affects the ability of users to connect to the Analysis server, which data they can access, and whether they have read or read/write access.

End-user security relies on the definition of user accounts and groups in Microsoft Windows NT® 4.0 or Windows® 2000 and on the creation of Microsoft SQL Server™ Analysis Services security roles, each specifying a set of users and groups. The process of implementing end-user security is summarized in the following table.

Phase	Description	For more information
1	Review and revise Windows NT 4.0 or Windows 2000 user accounts and groups in accordance with the various access requirements of your end users.	User Accounts and Groups
2	Create security roles and assign each role to the cubes or data mining models that the users in the role are permitted to access.	Database, Cube, and Mining Model Roles and Creating Security Roles
3	Define each role assigned to a cube or mining model. Each role's definition can vary for each cube or mining model to which it is assigned.	Levels of End-User Security and Creating Security Roles

User Accounts and Groups

Before you create roles in Microsoft® SQL Server™ 2000 Analysis Services, you must create user accounts and groups in User Manager in Microsoft Windows NT® 4.0 (or Computer Management in Windows® 2000). To be created, a role must contain at least one user account or group. A user account or group cannot be added to a role until after it is created in Windows NT 4.0 or Windows 2000.

It is usually best to finalize and implement a design for the memberships of groups before you create roles in Analysis Services. This approach can reduce the amount of required role maintenance.

If you are using NTLM Security Support Provider as your authentication method, all user accounts and groups that are to be granted access to cubes on an Analysis server must be in the same domain as the user account under which the Analysis server was installed or in a trusted domain. User accounts and groups in other domains cannot connect to the Analysis server.

Database, Cube, and Mining Model Roles

A role (also called a security role) defines a set of Microsoft® Windows NT® 4.0 or Windows® 2000 user accounts and groups with the same access to Microsoft SQL Server™ 2000 Analysis Services data. Roles are used to implement end user security by controlling access to data on the Analysis server by users connected with client applications. Analysis Services includes three types of roles: database role, cube role, and mining model role.

A database role can be assigned to multiple cubes or mining models in the database, thereby granting users of the role access to these cubes or mining models. Such an assignment creates a cube role or a mining model role with the same name as the database role. A database role provides defaults for cube or mining model roles of the same name. Although in a database role you can specify the type and scope of access to dimension members for cubes, this access is not actually granted until the database role is assigned to a cube. Database roles are defined at the Analysis Services database level, and are maintained in Database Role Manager.

By default, a database role specifies only read access and does not limit the dimension members or cube cells visible to end users. After such a database role is assigned to a cube, users in the role can view the entire cube. However, in both database roles and cube roles, you can specify read/write access and limit the dimension members that are visible and updatable. In cube roles you can limit the cube cells that are visible and updatable. On the other hand, mining model roles provide read-only access to model content.

A cube role applies to a single cube. Defaults in a cube role are derived from the database role of the same name, but some of these defaults can be overridden in the cube role. A cube role contains additional options, such as cell security, that are not contained in a database role. Cube roles are created at the cube level when a database role is assigned to a cube, and they are maintained in Cube Role Manager.

In cube roles, you can indicate whether end users in the role can drill through to a cell's source data. This capability also requires that drillthrough is enabled for the cube or at least one of its partitions. For more information, see Specifying

Drillthrough Options.

A mining model role applies to a single mining model. Default memberships in a mining model role are derived from the database role of the same name, but the default membership can be overridden in the mining model role. Mining model roles are created at the model level when a database role is assigned to a model, and they are maintained in Mining Model Role Manager.

An end user may be included in multiple roles on an Analysis server. In this case, the user has the combined access specified in these roles. If any one of the roles provides the user access to an object, the user has access to it. Exceptions are custom rules in dimension security. Not all combinations of custom rules from multiple roles can be resolved. For more information, see Multiple Dimension Custom Rules Applied to an End User.

The security enforcement provided by roles must be preceded by successful authentication of an end user as he or she connects to the Analysis server with a client application. If authentication is not successful, the user will not be able to access data on the Analysis server regardless of his or her membership in roles on that server and the definitions of those roles. For more information, see Server Security and Authentication.

See Also

Creating Security Roles

Cube Role Manager

Database Role Manager

Mining Model Role Manager

Levels of End-User Security

End-user security can be enforced at several levels of detail. These levels are summarized in the following table, sorted from least detailed to most detailed.

Level	Optional/ required	Description	For more information
Server	Required	Controls whether an end user can connect to an Analysis server with a client application.	Server Security and Authentication
Database	Required	Determines which Microsoft® SQL Server™ 2000 Analysis Services databases a connected end user can view.	Database Security
Cube/Mining Model	Required	Determines which cubes or mining models a connected end user can view.	Cube Security
Dimension member	Optional	Limits the dimension members that a connected end user can view. It can provide read/write access to write-enabled dimensions and limit the members that an end user can update.	Dimension Security
Cell	Optional	Limits the cube cells that a connected end user can view. It can provide read/write access to write-enabled cubes and limit the cells that an end user can update.	Cell Security

Server Security and Authentication

End-user security at the Analysis server level is controlled by authentication. To successfully connect to an Analysis server, an end user must be successfully authenticated on that Analysis server. Only after successful authentication are the roles on the Analysis server evaluated to determine the types and scope of access the end user has to objects on the Analysis server.

Authentication can be accomplished by various methods. The available methods depend in part on the way the end user attempts to connect to the Analysis server. If the end user tries to connect directly to the Analysis server, one set of authentication methods is available. If the end user tries to connect to the Analysis server through Internet Information Services (IIS), another set of authentication methods is available. For more information, see Authentication of Direct Connections and Authentication of Connections.

Authentication Methods

Microsoft® SQL Server™ 2000 Analysis Services supports three authentication providers:

- NTLM protocol (Microsoft Windows® authentication)
- Kerberos
- Negotiate

To connect to Analysis Services using one of these security providers, use the Security Support Provider Interface (SSPI) property. For more information, see SSPI Property.

For more information, see your Windows 2000 Security documentation.

Authentication of Direct Connections

The Analysis server authenticates end users when they attempt to connect directly to the server. These connections are characterized by:

- Connection strings containing a **Data Source** property value equivalent to an Analysis server name.

- Use of Transport Control Protocol/Internet Protocol (TCP/IP).

When an end user attempts to connect directly to an Analysis server, Microsoft® SQL Server™ 2000 Analysis Services attempts to authenticate the end user based on the credentials the end user was granted in the operating system when the end user logged on to the domain. Analysis Services automatically detects a connecting end user's credentials. If, in the connection string, the end user specifies a user name and password that is different from his or her logon user name and password, the specified user name and password are ignored. If the end user's credentials allow the end user to access the Analysis server computer from the network, authentication on the Analysis server is successful, and the end user is allowed to connect to the Analysis server. If the end user's credentials do not allow the end user to access the Analysis server computer from the network, authentication on the Analysis server is unsuccessful, and the end user is not allowed to connect to the Analysis server.

For authentication, Analysis Services uses Security Support Provider Interface (SSPI) as the interface to Microsoft Windows NT® 4.0 or Windows® 2000 security. Analysis Services supports Kerberos, NTLM Security Support Provider, and other providers that use SSPI. You can select the provider by setting the **SSPI** property in the connection string. For more information, see SSPI Property.

If the provider is NTLM Security Support Provider, access to an Analysis server requires an end user to be a member of the same domain as the user account under which the Analysis server was installed, or to be a member of a trusted domain. An end user is denied access if the end user's account cannot be authenticated against one of these domains.

Another type of connection, which is through Internet Information Services (IIS), can also be attempted. For more information, see Authentication of Connections.

See Also

Registering Servers

Connected to Analysis Services

Authentication of Connections

The Analysis server authenticates end users when they attempt to connect to an Analysis server through Microsoft® Internet Information Services (IIS). These connections are characterized by:

- Connection strings containing a **Data Source** property value equivalent to a URL.
- Use of Hypertext Transfer Protocol (HTTP).

Note Connections through HTTP and IIS from Analysis Manager or from client applications connecting through PivotTable® Service are available only if you install Analysis Services for Microsoft SQL Server™ 2000 Enterprise Edition.

Direct connections can also be attempted. For more information, see Authentication of Direct Connections.

When an end user attempts to connect to an Analysis server through IIS, Analysis Services relies on the authentication on IIS. If authentication on IIS is successful, authentication on the Analysis server is successful, and the end user is allowed to connect to the Analysis server. If authentication on IIS is unsuccessful, authentication on the Analysis server is unsuccessful, and the end user is not allowed to connect to the Analysis server.

IIS provides several authentication methods. For example, a user logon and password can be used for Basic authentication for HTTP or secure HTTP connections. Other methods can be integrated with roles in Analysis Services. For more information, see the IIS documentation.

See Also

Registering Servers

Connected to Analysis Services

Connecting Using HTTP

Database Security

Database security is controlled using database roles. After an end user successfully connects to an Analysis server, database roles on that server are searched for the end user's user name.

If the user name is found in a database role, the end user can view that database's name and a list of cubes (including virtual and linked cubes) in that database. However, the end user can access only those cubes to which the database role has been assigned.

If the user name is not found in the database roles, the end user cannot view or access any objects on the server.

Before you grant end users access to cubes in a database, you must grant them database access by including them in a database role.

See Also

Creating Database Roles

Cube Security

Database, Cube, and Mining Model Roles

Database Role Manager

Cube Security

The type and scope of access to a cube by end users in a cube role is determined by the settings in the cube role. An end user can access only those cubes that are assigned a role containing that end user's user name.

A database role provides defaults for the cube roles of the same name, but some of these defaults can be overridden in the cube roles. After a database role is created, it can be assigned to any cube (including virtual and linked cubes) in the database. This action grants the end users in the database role access to the cube and creates a cube role with the same name as the database role. Database roles are assigned to cubes in the **Cubes** tab of the **Database Role** dialog box or in Cube Role Manager.

If a cube role does not specify restrictions on dimension members, end users in the cube role can view all members in the associated cube. If a dimension has been write-enabled, and the cube role has been granted read/write access to the dimension, the end users can also update members in the dimension. However, a database role or cube role can specify that some members can be viewed and updated and others cannot. For more information, see Dimension Security.

Similarly, by default, end users in a cube role can view all cells in the associated cube. If the cube has been write-enabled, and the cube role has been granted read/write access to the cube, the end users can also update cube cells. However, a cube role can specify that some cells can be viewed and updated and others cannot. For more information, see Cell Security.

By default, end users in a cube role cannot drill through to any of the cube cells' source data. However, in a cube role you can grant this ability. If you grant this ability, you must enable drillthrough for the cube or for at least one of its partitions.

See Also

Creating Security Roles

Database, Cube, and Mining Model Roles

Specifying Drillthrough Options

Dimension Security

In a database role or cube role, you can implement dimension security to specify the dimension members that end users in the role can view as they browse cubes. You can also grant read/write access to a write-enabled dimension and specify the members that end users in the role can update.

Dimension security is optional. If you do not specify dimension security, end users see all dimension members in the cubes they are authorized to access. If a dimension is write-enabled, they cannot update members.

You can specify dimension security at both the database and cube levels. In a database role, for a shared dimension you can define specifications that apply to all of the database's cubes that include that dimension. These specifications provide defaults for the cube roles with the same name as the database role. In a cube role, you can override these specifications for a specific cube.

Note Unlike updates to cube cells, updates to dimension members are recorded directly in the source table. These updates can include additions, deletions, renames, and moves.

In a database role, dimension security is defined in the **Dimensions** tab of the **Database Role** dialog box. In a cube role, it is defined in the **Dimensions** tab of the **Cube Role** dialog box.

Permissions and Rules

You can set permissions and rules for groups you define in Microsoft® Windows NT® 4.0 or Windows® 2000 to manage dimension security. In addition, you can specify individual members and groups of members that can be updated and that cannot be updated. For more information, see Custom Rules in Dimension Security.

When you specify dimension security within a role, you can define permissions for each dimension.

Permission	Description

Read	Determines which members are viewable. This permission affects the size of the visible cube because it limits the members that are displayed.
Read/write	Determines which members are updatable. You can define and grant this permission only if the dimension has been write-enabled. If you grant this permission and the dimension is later write-disabled, this permission is disabled, and end users cannot update the dimension's members.

Members specified in the read/write permission are also viewable. Therefore, if the read/write permission includes members that are not in the read permission, the read/write permission also affects the size of the visible cube.

For the read permission, you can select one of the following rules.

Rule	Description
Unrestricted	End users can view all members. This rule is the default.
Fully Restricted	End users cannot view members. When they browse a cube that includes the dimension, they do not see it.
Custom	This rule provides the most flexibility. Specify **Top level**, which indicates the topmost level that can be viewed.

For the read/write permission, you can select one of the following rules.

Rule	Description
Unrestricted	End users can update all members. This rule is available only if the read permission's rule is Unrestricted.
Fully Restricted	End users cannot update members. This rule is the default. This rule is available only if the read permission's rule is Unrestricted or Fully Restricted.
Custom	This rule provides the most flexibility. This rule is available only if the read permission's rule is Unrestricted or Custom. Specify **Top level**, which indicates the topmost dimension level that can be updated, or **Bottom level**, which indicates the bottommost dimension level

	that can be updated.

See Also

Creating Security Roles

Custom Rules in Dimension Security

Write-Enabled Dimensions

Custom Rules in Dimension Security

In a database role or cube role, by defining a custom rule for a dimension, you can specify which dimension members can be accessed by end users in the role. To do this, you can select the accessible levels, or specify the accessible members, or use these methods in combination. You can also specify options for visual totals and select a default member.

Level Selections

For each dimension you can specify a range of dimension levels that can be accessed by selecting one or both of the following:

- Top level

  Indicates the topmost level that can be accessed. Levels above the top level cannot be accessed. By default, top level is the dimension's highest level.

- Bottom level

  Indicates the bottommost level that can be accessed. Levels below the bottom level cannot be accessed. By default, bottom level is the dimension's lowest level.

  Examples 1, 2, and 3 illustrate this concept.

Note Although members above the selected top level cannot be accessed, if a client application reveals the names of members' ancestors above the top level (for example, displays fully-qualified member names), end users might deduce cell values for members above the top level.

Member Specifications

You can specify the members that can be accessed by allowing access to some members and denying access to others.

You can combine top level and bottom level selections with member specifications. However, you cannot allow access to members above the top level or below the bottom level.

Descendants of a specified member share the same access. For example, if you explicitly allow access to England, you also implicitly allow access to London, Manchester, and England's other descendants. However, there are three exceptions:

- Access to a member is allowed, but a bottom level is selected. Descendants below the bottom level cannot be accessed.
- Access to a member is allowed, but access to a descendant is denied. For example, you allow access to England but deny access to London. In this case, England's descendants, except London and its descendants, can be accessed.
- Access to a member is denied, but access to a descendant is allowed. In this case, the member you allowed access to and its ancestors up to and including the denied member can be accessed. The descendants of the allowed member can also be accessed. However, no other descendants of the denied member can be accessed. For example, you deny access to Europe but allow access to London. London, England, and Europe can be accessed, London's descendants can be accessed, but other descendants of Europe cannot be accessed.

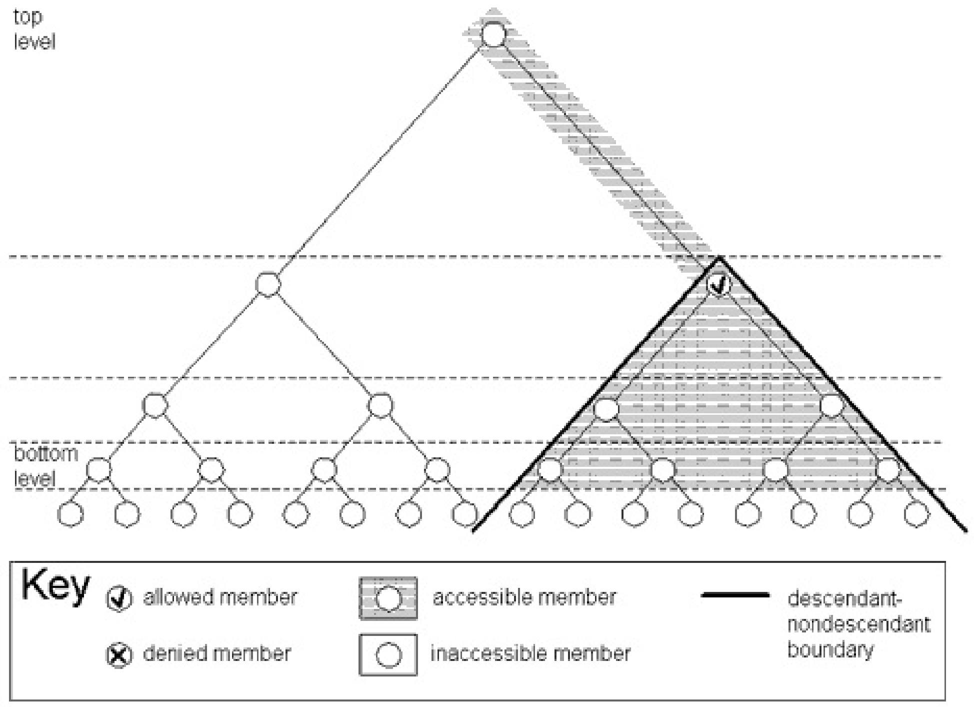
Ancestors of an explicitly allowed member can also be accessed unless they are above the top level. This is true even if the ancestor is explicitly denied.

In a member specification, you have three basic choices: allow only, deny only, or allow and deny.

Allow Only (Member Specifications)

The members you explicitly allow access to and their descendants and ancestors can be accessed unless they are below the bottom level. The only nondescendants that can be accessed are the allowed members' ancestors at or below the top level.

The following diagram shows a member hierarchy with access explicitly allowed to a single member.

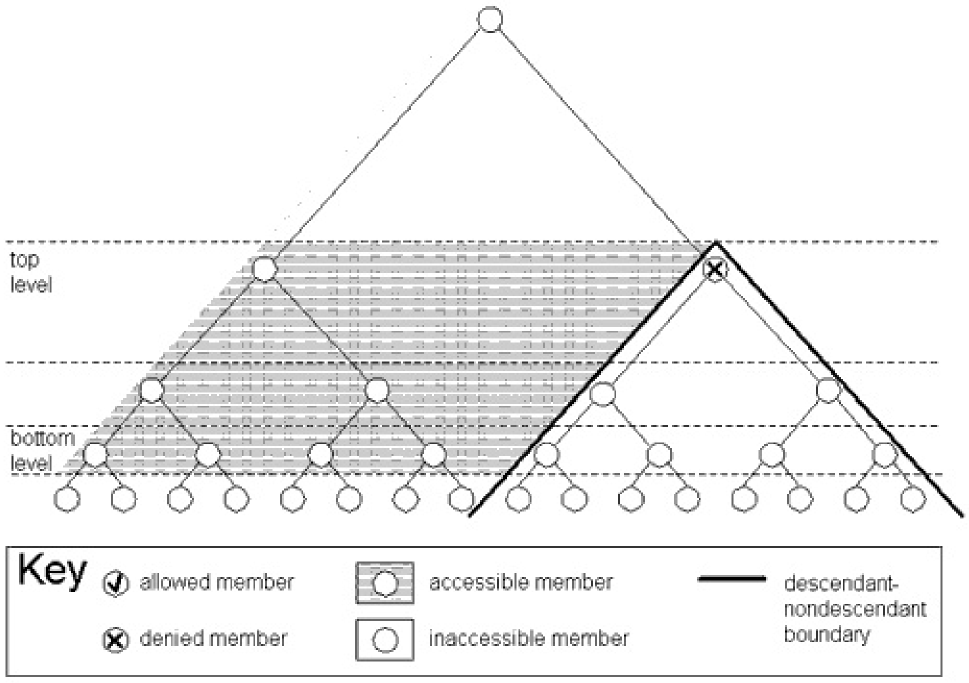


Examples 4 and 7 illustrate this concept.

Deny Only (Member Specifications)

The members you explicitly deny access to and their descendants cannot be accessed. Nondescendants can be accessed unless they are above the top level or below the bottom level.

The following diagram shows a member hierarchy with access explicitly denied to a single member.



Examples 5 and 8 illustrate this concept.

Allow and Deny (Member Specifications)

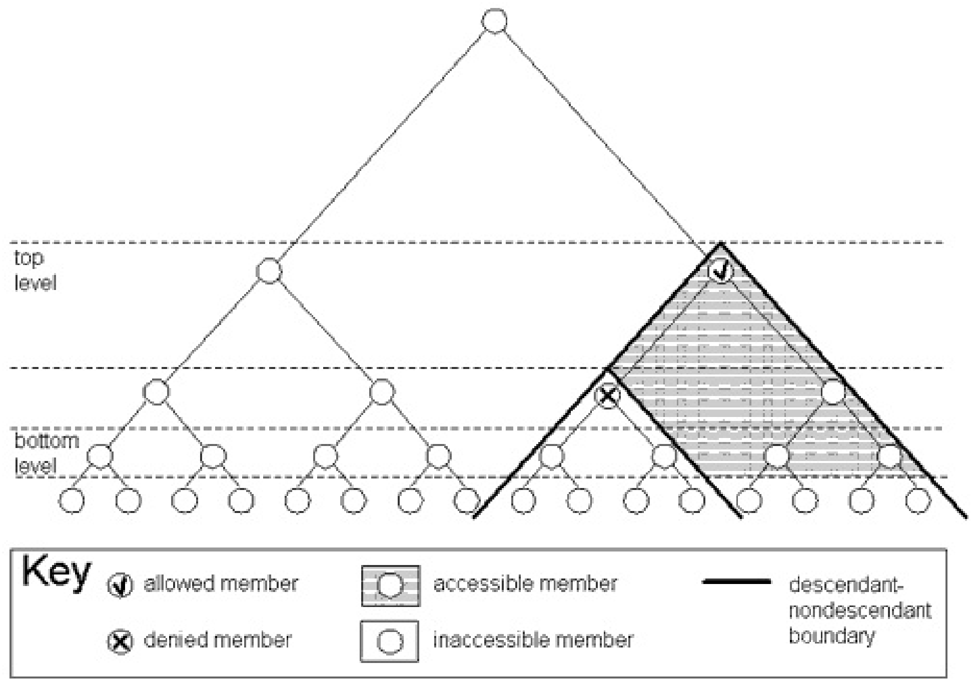
There are two common relationships between allowed members and denied members:

- All of the explicitly denied members are descendants of the explicitly allowed members.

- All of the explicitly allowed members are descendants of the explicitly denied members.

If all of the explicitly denied members are descendents of the explicitly allowed members, the members you explicitly allow access to and their descendants can be accessed with the following exceptions: (1) descendants you explicitly deny access to and their descendants cannot be accessed; and (2) descendants below the bottom level cannot be accessed. The only nondescendants of the explicitly allowed members that can be accessed are the allowed members' ancestors at or below the top level.

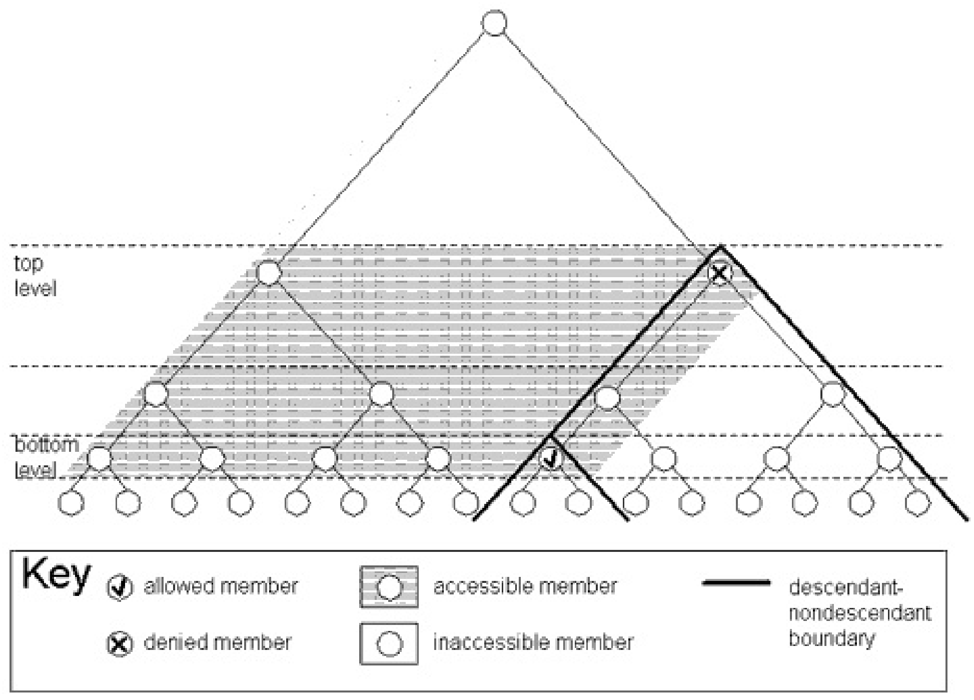
The following diagram shows a member hierarchy with access explicitly allowed to a single member and access explicitly denied to one of its descendants.



Examples 6 and 9 illustrate this concept.

If all of the explicitly allowed members are descendents of the explicitly denied members, the members you explicitly deny access to and their descendants cannot be accessed with the following exceptions: (1) descendants you explicitly allow access to and their descendants can be accessed unless they are below the bottom level; and (2) descendants you explicitly allow access to and their ancestors up to and including the denied members can be accessed. Nondescendants of the explicitly denied members can be accessed if they are between the top level and bottom level.

The following diagram shows a member hierarchy with access explicitly denied to a single member and access explicitly allowed to one of its descendants.



Example 10 illustrates this concept.

Note This relationship between allowed and denied members cannot be defined in the **Basic** tab of the **Custom Dimension Security** dialog box. You must use the **Advanced** tab.

Other relationships between allowed members and denied members are possible.

Visual Totals

For each dimension you can specify options for visual totals. These options determine whether displayed, aggregated cell values are calculated according to all of a member's descendants or only the viewable descendants. In the first case, end users in the role see actual totals; in the second, they see visual totals. A third option is available to display visual totals at and above a specified level, but display actual totals below it. This option requires an expression in Multidimensional Expressions (MDX). This expression must name the level or resolve to the level at and above which visual totals are displayed. Example 11 illustrates this concept.

Note Visual totals cannot be enabled for a cube that contains a distinct count measure. For more information, see Using Aggregate Functions.

By default, visual totals are disabled. In this case, displayed, aggregated cell values are calculated according to all of a member's descendants, regardless of whether they are viewable. If some members are not viewable, the default setting can cause some cell values to appear incorrect to end users.

IMPORTANT The default setting (that is, visual totals are disabled) creates security exposures if it allows end users to deduce values for members to which they are denied access. Examples 4, 6, 8, and 9 illustrate this concept.

Default Member

For each dimension you can select a default member. The default member affects the datasets returned by queries on cubes that include the dimension. When the dimension is not displayed on an axis, by default the dataset is filtered (that is, sliced) using the default member. Example 12 illustrates this concept.

If you do not select a default member, the default member is determined by the dimension's **Default Member** property, which is accessed in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

Custom Dimension Security Dialog Box

Custom rules for dimension security are defined in the **Custom Dimension Security** dialog box. To indicate the levels and members that can be accessed, you can choose from two methods:

- Select items by using drop-down lists, a member tree, and other elements of the dialog box. This method is used in the **Basic** tab.

- Write MDX. With this method, you can implement all techniques permitted by the preceding method plus a few, infrequently used techniques. This method is used in the **Advanced** tab.

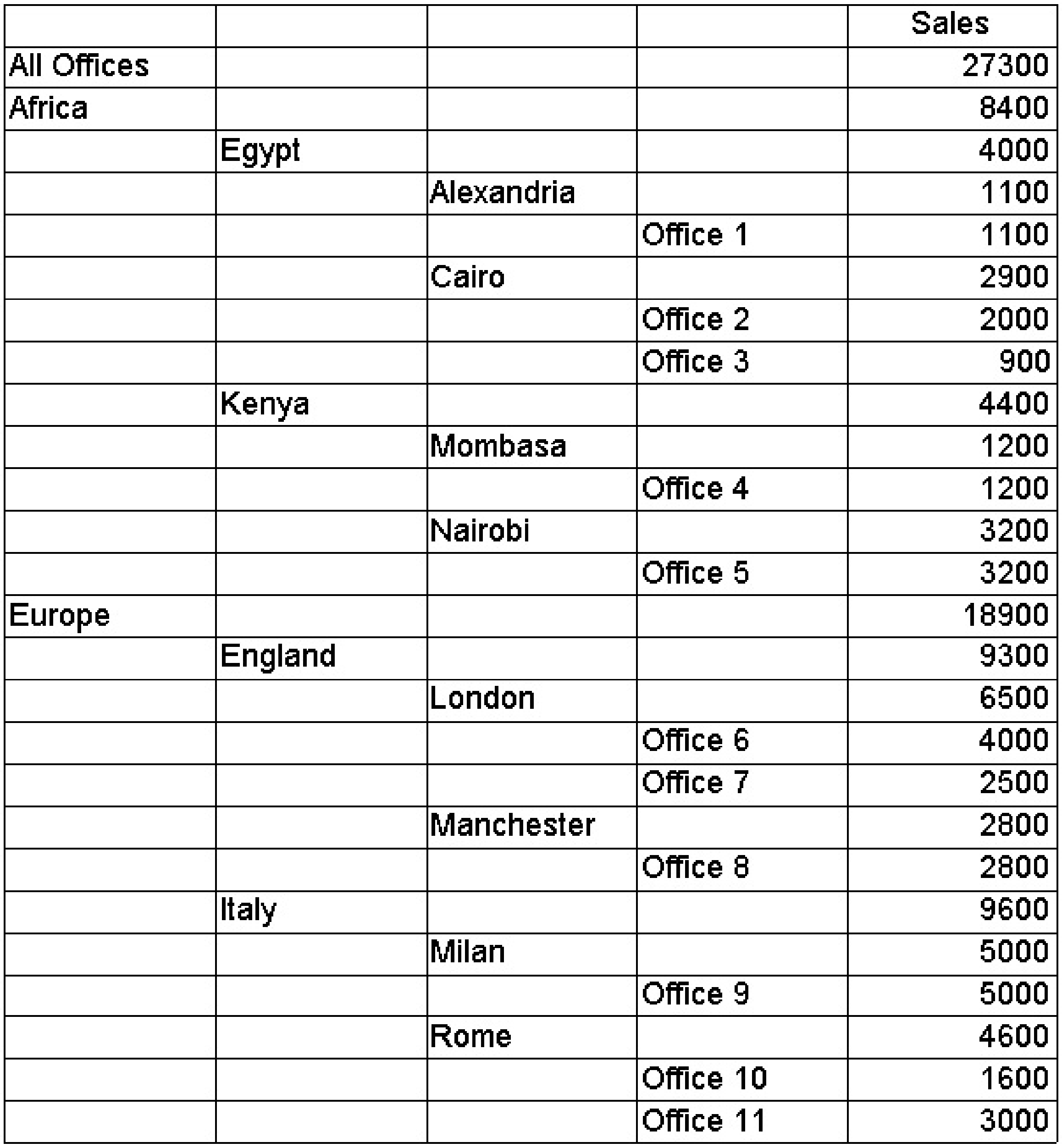
Options for visual totals and the default member are specified in the **Common** tab.

Example Custom Rules in Dimension Security

The examples are for an Offices dimension defined with the following levels:

- (All), containing only the All Offices member
- Continent
- Country
- City
- Office

Before dimension security is implemented, a dataset returned from a cube with this dimension looks like this. (The dimension is fully expanded to show all members.)



				Sales
All Offices				27300
Africa				8400
	Egypt			4000
		Alexandria		1100
			Office 1	1100
		Cairo		2900
			Office 2	2000
			Office 3	900
	Kenya			4400
		Mombasa		1200
			Office 4	1200
		Nairobi		3200
			Office 5	3200
Europe				18900
	England			9300
		London		6500
			Office 6	4000
			Office 7	2500
		Manchester		2800
			Office 8	2800
	Italy			9600
		Milan		5000
			Office 9	5000
		Rome		4600
			Office 10	1600
			Office 11	3000

Each example defines alternative dimension security for this dimension. Each example includes one or more MDX expressions in a custom rule for a read permission. The result on the preceding dataset is shown for each example except Example 12, which includes only an expression for a default member.

The examples are applied to read permissions so that the effects of the MDX expressions on the Offices dimension can be visually demonstrated. However, the examples are also applicable to read/write permissions. That is, if an example's expressions were used in a read/write permission, they would allow and deny access to the same members as in the read permission. Exceptions are

Examples 11 and 12, which demonstrate options that are defined once per dimension and cannot vary between the read and read/write permissions.

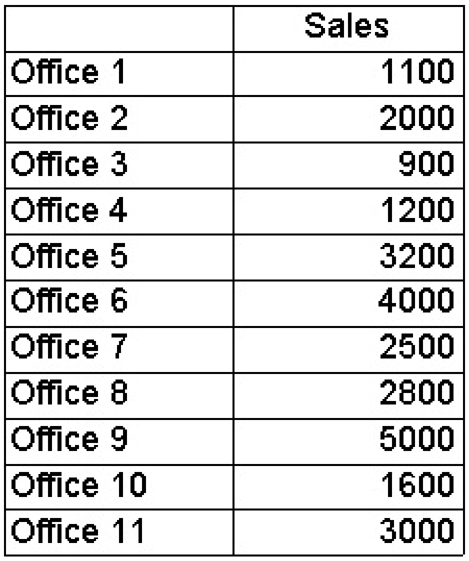
Example 1

This example includes only a top level selection.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box:
      [Offices].[Office]
    - **Bottom Level** box: no expression specified. However, by default the following expression is supplied by Microsoft® SQL Server™ 2000 Analysis Services to represent the dimension's lowest level:
      [Offices].[Office]
    - **Allowed Members** box: no expression specified.
    - **Denied Members** box: no expression specified.
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)
    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



	Sales
Office 1	1100
Office 2	2000
Office 3	900
Office 4	1200
Office 5	3200
Office 6	4000
Office 7	2500
Office 8	2800
Office 9	5000
Office 10	1600
Office 11	3000

Example 2

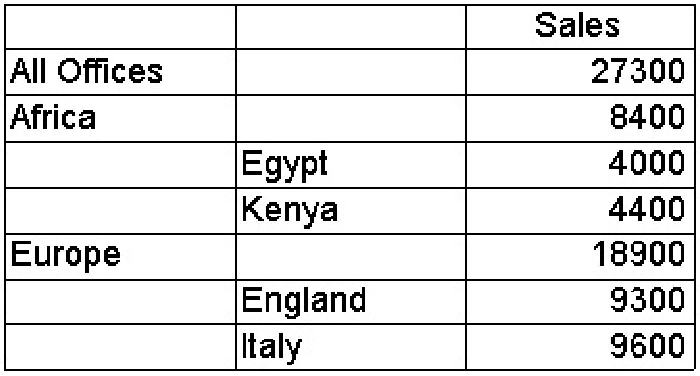
This example includes only a bottom level selection.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
  - **Top Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's highest level:
    [Offices].[(All)]
  - **Bottom Level** box:
    [Offices].[Country]
  - **Allowed Members** box: no expression specified.
  - **Denied Members** box: no expression specified.
- **Common** tab
  - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)

    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



		Sales
All Offices		27300
Africa		8400
	Egypt	4000
	Kenya	4400
Europe		18900
	England	9300
	Italy	9600

Example 3

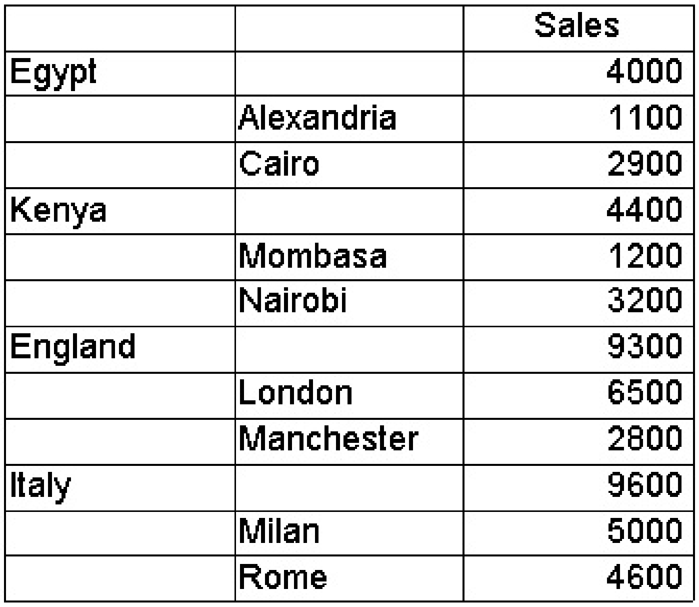
This example includes only a top level selection and bottom level selection.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box:
      [Offices].[Country]
    - **Bottom Level** box:
      [Offices].[City]
    - **Allowed Members** box: no expression specified.
    - **Denied Members** box: no expression specified.
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)

        - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



		Sales
Egypt		4000
	Alexandria	1100
	Cairo	2900
Kenya		4400
	Mombasa	1200
	Nairobi	3200
England		9300
	London	6500
	Manchester	2800
Italy		9600
	Milan	5000
	Rome	4600

Example 4

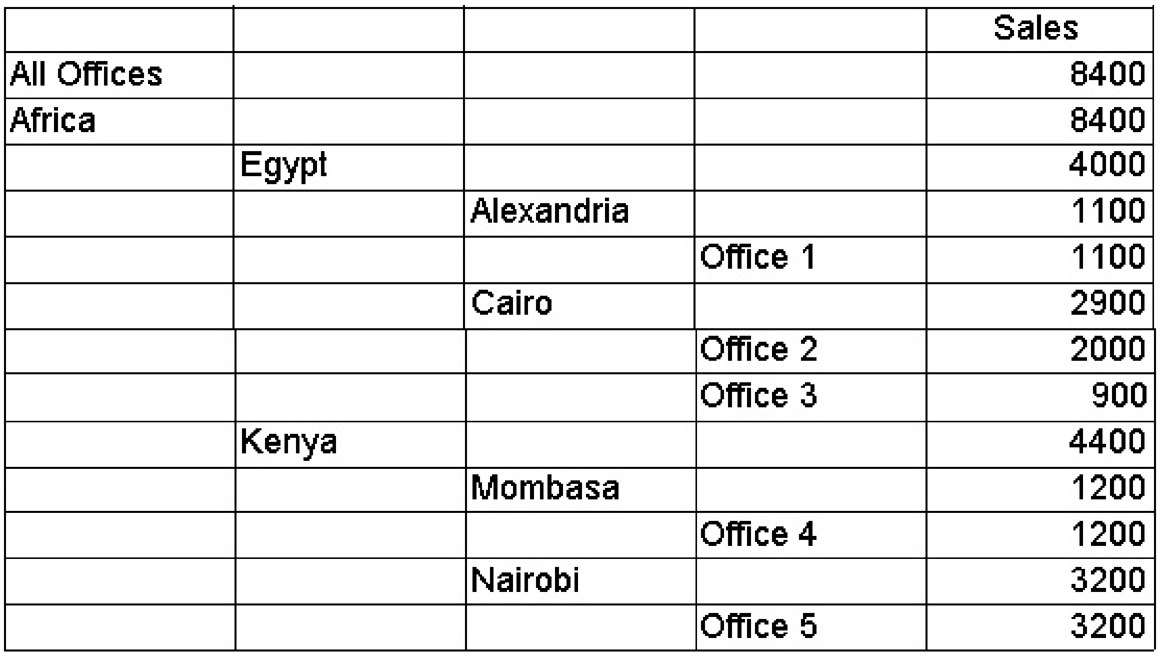
This example allows access to some members but does not include a top level selection or bottom level selection.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's highest level:
      [Offices].[(All)]
    - **Bottom Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's lowest level:
      [Offices].[Office]

    - **Allowed Members** box: {[Offices].[All Offices].[Africa]}
    - **Denied Members** box: no expression specified.
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)
    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



				Sales
All Offices				8400
Africa				8400
	Egypt			4000
		Alexandria		1100
			Office 1	1100
		Cairo		2900
			Office 2	2000
			Office 3	900
	Kenya			4400
		Mombasa		1200
			Office 4	1200
		Nairobi		3200
			Office 5	3200

Note Because visual totals are enabled, the displayed Sales for All Offices is 8400, although the actual Sales is 27300. If visual totals were disabled (the default), a security exposure would exist because end users could calculate the Europe Sales by subtracting the Africa Sales from the actual All Offices Sales.

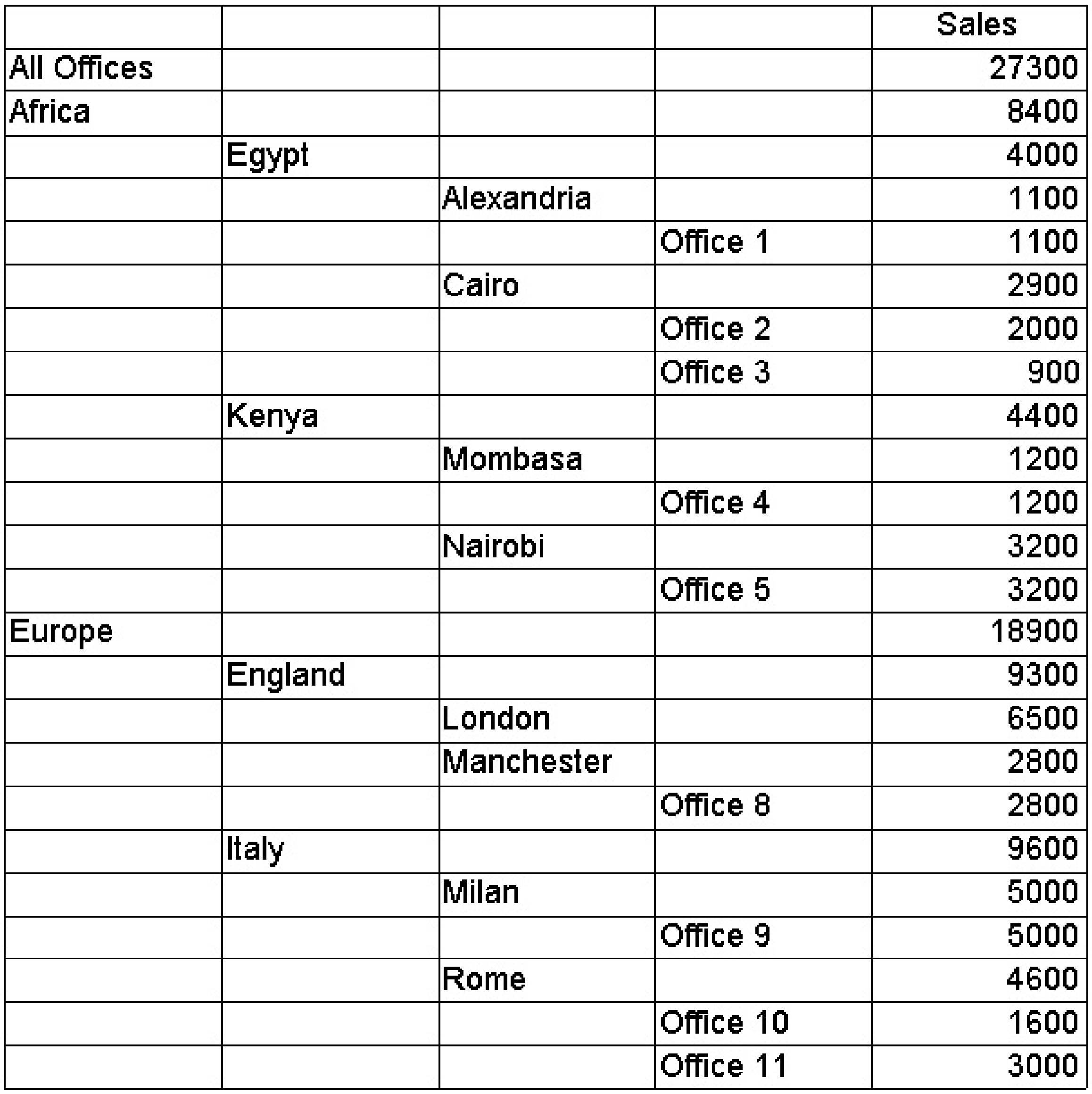
Example 5

This example denies access to some members but does not include a top level selection or bottom level selection.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's highest level:
      [Offices].[(All)]
    - **Bottom Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's lowest level:
      [Offices].[Office]
    - **Allowed Members** box: no expression specified.
    - **Denied Members** box:
      {[Offices].[All Offices].[Europe].[England].[London].[
      [Offices].[All Offices].[Europe].[England].[London].[C
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)
    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



				Sales
All Offices				27300
Africa				8400
	Egypt			4000
		Alexandria		1100
			Office 1	1100
		Cairo		2900
			Office 2	2000
			Office 3	900
	Kenya			4400
		Mombasa		1200
			Office 4	1200
		Nairobi		3200
			Office 5	3200
Europe				18900
	England			9300
		London		6500
		Manchester		2800
			Office 8	2800
	Italy			9600
		Milan		5000
			Office 9	5000
		Rome		4600
			Office 10	1600
			Office 11	3000

Example 6

This example allows access to some members but denies access to a subset of their descendants. It does not include a top level selection or bottom level selection.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
  - **Top Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to

    represent the dimension's highest level:
    [Offices].[(All)]

    - **Bottom Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's lowest level:
      [Offices].[Office]

    - **Allowed Members** box:
      {[Offices].[All Offices].[Europe].[Italy]}

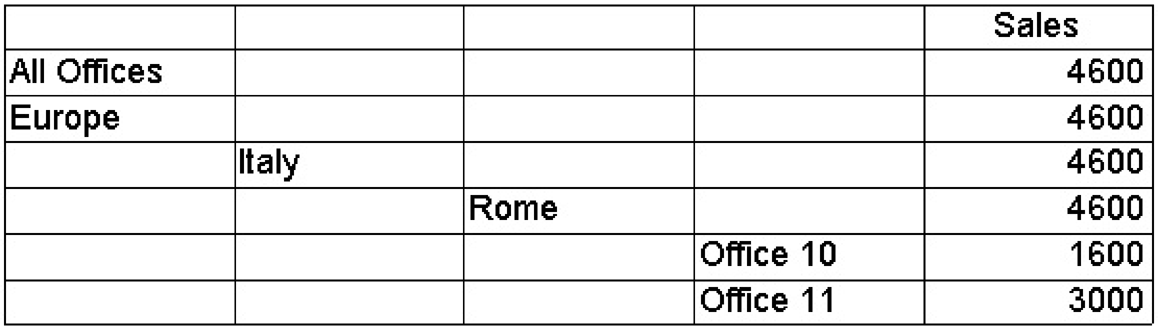
    - **Denied Members** box:
      {[Offices].[All Offices].[Europe].[Italy].[Milan]}

- **Common** tab

    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)

    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



				Sales
All Offices				4600
Europe				4600
	Italy			4600
		Rome		4600
			Office 10	1600
			Office 11	3000

Note Because visual totals are enabled, the displayed Sales for Italy is 4600, although the actual Sales is 9600. The displayed Sales for Europe is 4600, although the actual Sales is 18900. The displayed Sales for All Offices is 4600, although the actual Sales is 27300. If visual totals were disabled (the default), security exposures would exist. For example, end users could calculate the Milan

Sales by subtracting the Rome Sales from the actual Italy Sales.

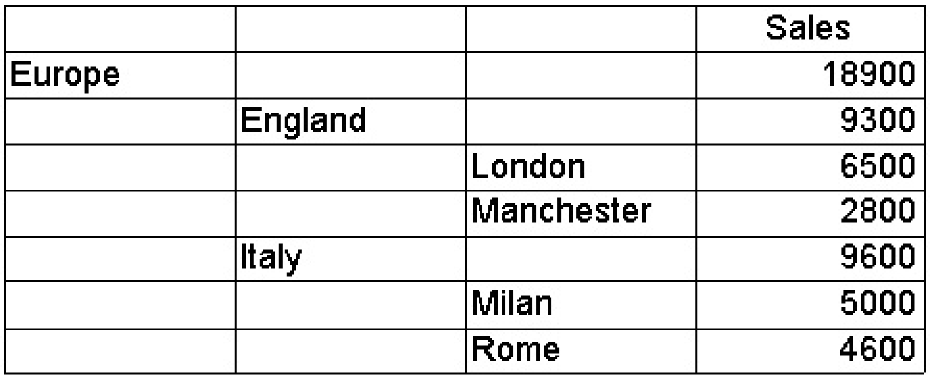
Example 7

This example includes a top level selection and bottom level selection and allows access to some members between the top level and bottom level.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
  - **Top Level** box:
    [Offices].[Continent]
  - **Bottom Level** box:
    [Offices].[City]
  - **Allowed Members** box:
    {[Offices].[All Offices].[Europe]}
  - **Denied Members** box: no expression specified.
- **Common** tab
  - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)
  - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



			Sales
Europe			18900
	England		9300
		London	6500
		Manchester	2800
	Italy		9600
		Milan	5000
		Rome	4600

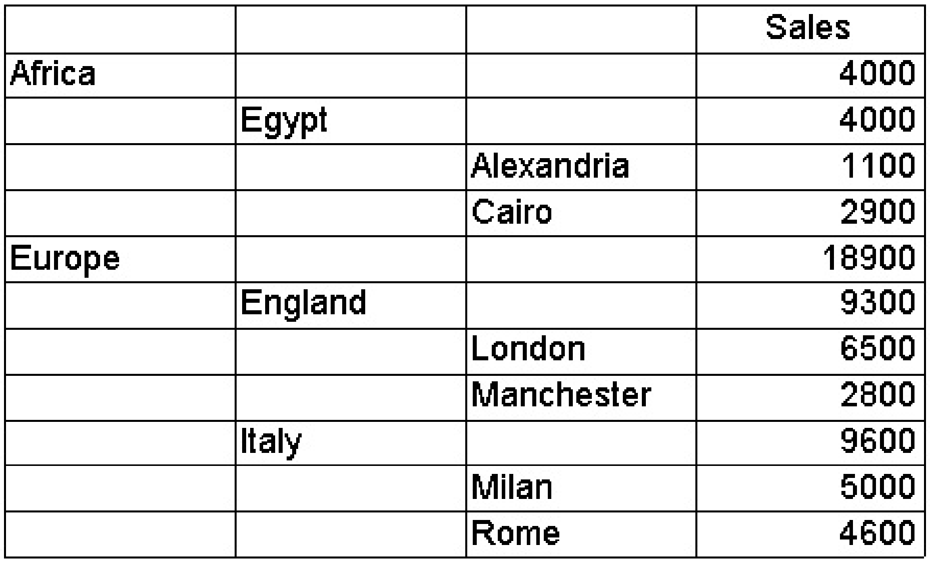
Example 8

This example includes a top level selection and bottom level selection but denies access to some members between the top level and bottom level.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box:
      [Offices].[Continent]
    - **Bottom Level** box:
      [Offices].[City]
    - **Allowed Members** box: no expression specified.
    - **Denied Members** box:
      {[Offices].[All Offices].[Africa].[Kenya]}
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.).
    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



			Sales
Africa			4000
	Egypt		4000
		Alexandria	1100
		Cairo	2900
Europe			18900
	England		9300
		London	6500
		Manchester	2800
	Italy		9600
		Milan	5000
		Rome	4600

Note Because visual totals are enabled, the displayed Sales for Africa is 4000, although the actual Sales is 8400. If visual totals were disabled (the default), a security exposure would exist because end users could calculate the Kenya Sales by subtracting the Egypt Sales from the actual Africa Sales.

Example 9

This example includes a top level selection and bottom level selection. It also allows access to some members between the top level and bottom level but denies access to a subset of their descendants.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

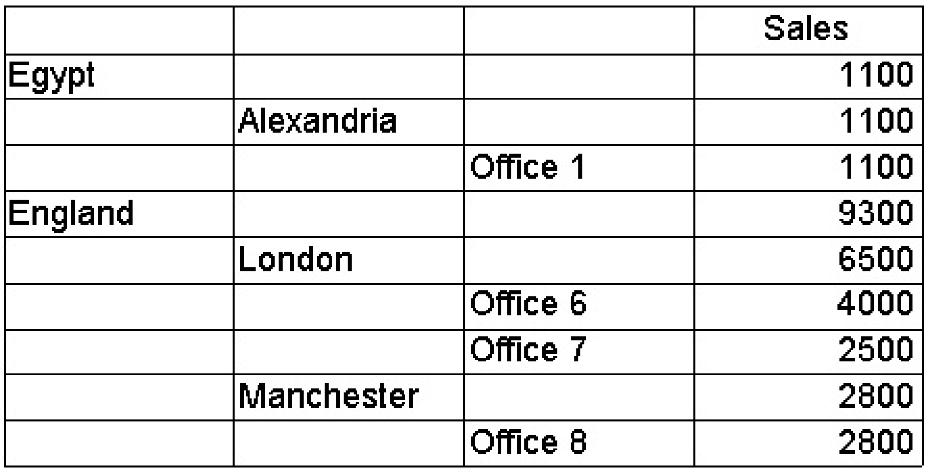
- **Advanced** tab
    - **Top Level** box:
      [Offices].[Country]
    - **Bottom Level** box:
      [Offices].[Office]
    - **Allowed Members** box:

    {[Offices].[All Offices].[Africa].[Egypt],
    [Offices].[All Offices].[Europe].[England]}

    - **Denied Members** box:
      {[Offices].[All Offices].[Africa].[Egypt].[Cairo]}

- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)
    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



			Sales
Egypt			1100
	Alexandria		1100
		Office 1	1100
England			9300
	London		6500
		Office 6	4000
		Office 7	2500
	Manchester		2800
		Office 8	2800

Note Because visual totals are enabled, the displayed Sales for Egypt is 1100, although the actual Sales is 4000. If visual totals were disabled (the default), a security exposure would exist because end users could calculate the Cairo Sales by subtracting the Alexandria Sales from the actual Egypt Sales.

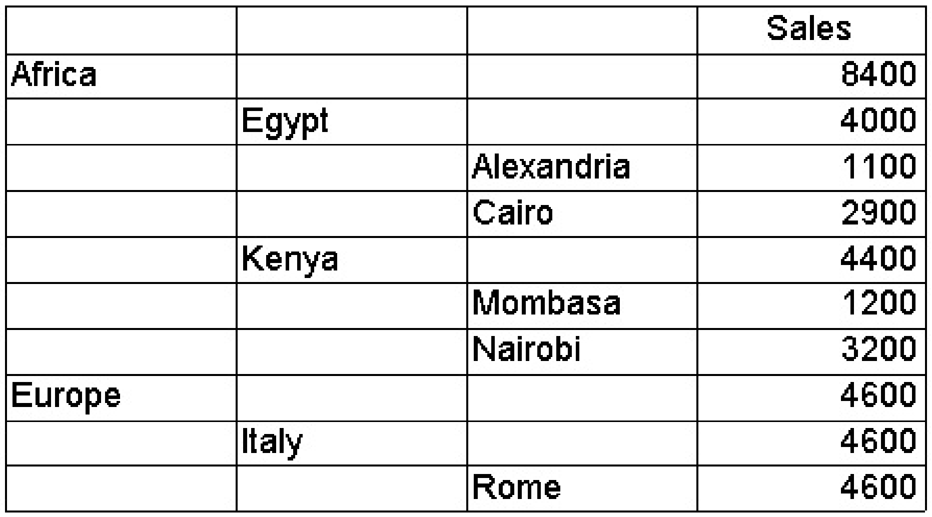
Example 10

This example includes a top level selection and bottom level selection. It also denies access to some members between the top level and bottom level but allows access to a subset of their descendants.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box:
      [Offices].[Continent]
    - **Bottom Level** box:
      [Offices].[City]
    - **Allowed Members** box:
      {[Offices].[All Offices].[Europe].[Italy].[Rome]}
    - **Denied Members** box:
      {[Offices].[All Offices].[Europe]}
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)
    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



			Sales
Africa			8400
	Egypt		4000
		Alexandria	1100
		Cairo	2900
	Kenya		4400
		Mombasa	1200
		Nairobi	3200
Europe			4600
	Italy		4600
		Rome	4600

Note Because visual totals are enabled, the displayed Sales for Europe is 4600, although the actual Sales is 18900. The displayed Sales for Italy is 4600, although the actual Sales is 9600.

Example 11

This example includes an MDX expression for a customized visual total.

The Europe Manager role is authorized to access values for the Europe member, all its descendants, the Alexandria member, and all its descendants. It is not authorized to access values for the Africa member. Allowing access to Alexandria implicitly allows access to Africa, but selecting the Continent level for visual totals ensures that the role cannot access actual values for Africa. As long as the role retains access to Europe and all its descendants, this visual totals selection will not impede the role's ability to view actual values for Europe.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
    - **Top Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's highest level:
      [Offices].[(All)]
    - **Bottom Level** box: no expression specified. However, by default the following expression is supplied by Analysis

    Services to represent the dimension's lowest level:
    [Offices].[Office]

    - **Allowed Members** box:
      {[Offices].[All Offices].[Europe], [Offices].[All Offices].[Africa].[Egypt].[Alexandria]}

    - **Denied Members** box: no expression specified.

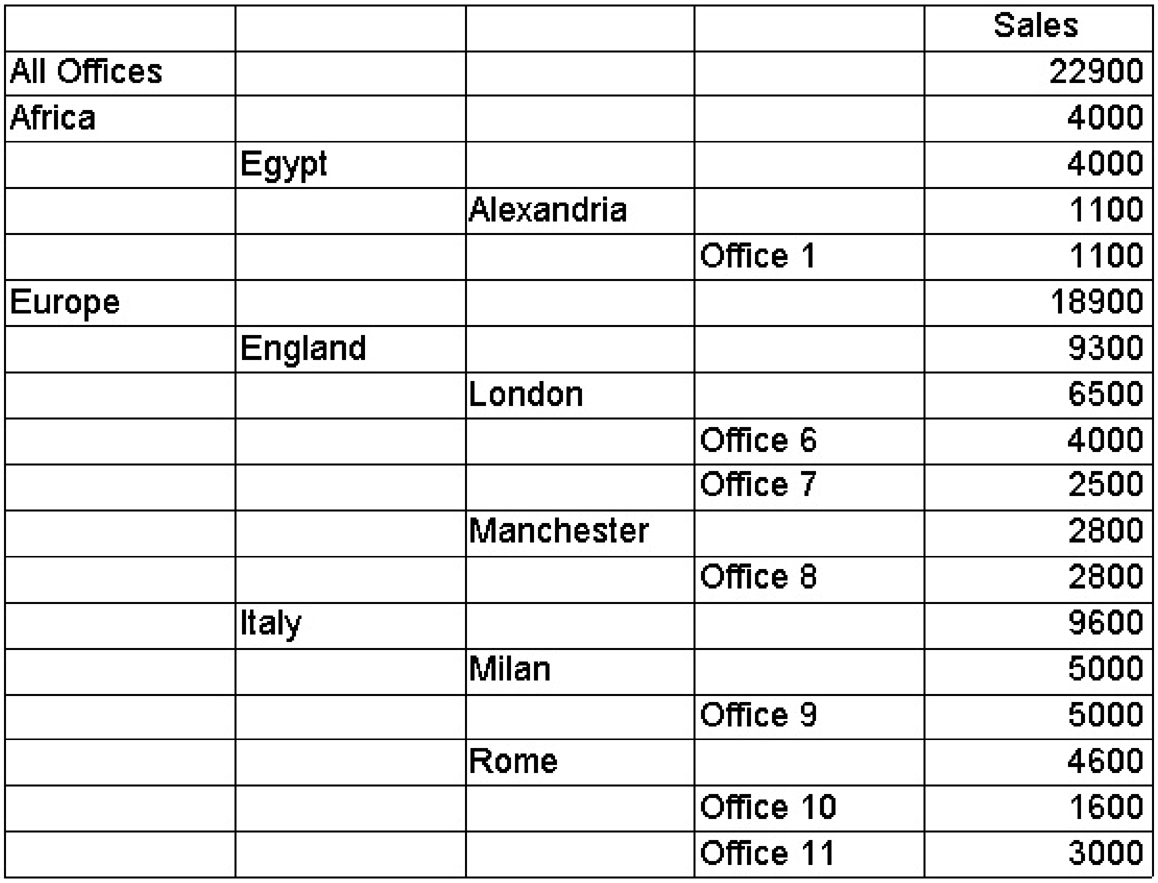
- **Common** tab

    - **Visual Totals** area:
      [Offices].[Continent]

      The selected option is **Custom - Show visual totals starting at the following level and above**.

    - **Default Member** area: no expression specified.

In a read permission, these expressions cause the example dataset to be modified as follows. (The following dimension is fully expanded to show all viewable members.)



				Sales
All Offices				22900
Africa				4000
	Egypt			4000
		Alexandria		1100
			Office 1	1100
Europe				18900
	England			9300
		London		6500
			Office 6	4000
			Office 7	2500
		Manchester		2800
			Office 8	2800
	Italy			9600
		Milan		5000
			Office 9	5000
		Rome		4600
			Office 10	1600
			Office 11	3000

Example 12

This example includes an MDX expression that specifies a default member. The default member is selected from the sample dimension.

The following MDX expressions are specified in the following boxes and areas of the **Custom Dimension Security** dialog box:

- **Advanced** tab
  - **Top Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's highest level:
    [Offices].[(All)]
  - **Bottom Level** box: no expression specified. However, by default the following expression is supplied by Analysis Services to represent the dimension's lowest level:
    [Offices].[Office]

    - **Allowed Members** box: no expression specified.

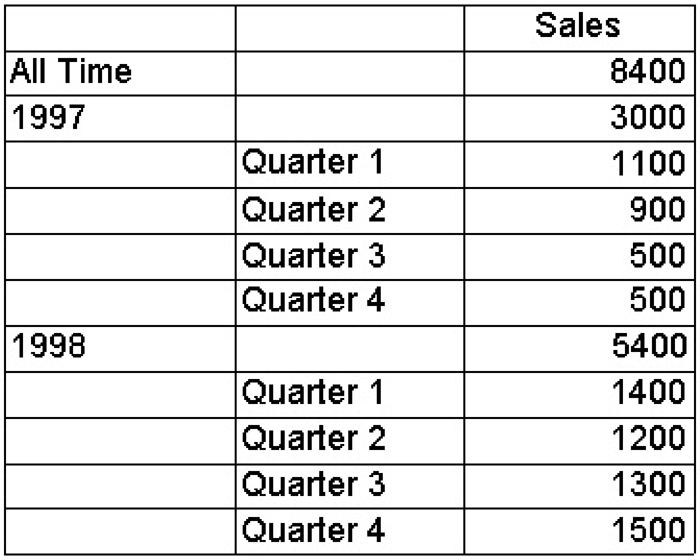
    - **Denied Members** box: no expression specified.
- **Common** tab
    - **Visual Totals** area: no expression specified, but visual totals are enabled. (That is, **Enable - Show visual totals** is selected.)

    - **Default Member** area:
      [Offices].[All Offices].[Africa]

      The **Define default member and specify using MDX** option is selected.

Whenever the Offices dimension is not projected on an axis, by default datasets are filtered (that is sliced) by the Africa member. Cell values in the dataset reflect Africa but not Europe.

For example, if Offices and Time are the only dimensions in the cube, and Sales is displayed with the Time dimension on the y-axis, the displayed Sales for the All Time member is 8400. The Sales for All Time and All Offices is 27300, but the dataset does not display this value because Europe values are excluded.



		Sales
All Time		8400
1997		3000
	Quarter 1	1100
	Quarter 2	900
	Quarter 3	500
	Quarter 4	500
1998		5400
	Quarter 1	1400
	Quarter 2	1200
	Quarter 3	1300
	Quarter 4	1500

See Also

Defining Custom Rules for Dimension Security

Multiple Dimension Custom Rules Applied to an End User

If, on an Analysis server, an end user is included in multiple cube roles that contain custom rules for dimension security, it is not always possible to resolve the combination of the custom rules. The easiest way to test whether and how the combination is resolved is to perform the following procedure.

To test the combination of multiple custom rules for dimension security

1. In the Analysis Manager tree pane, under the database that contains the cube roles, expand the Cubes folder.
2. Right-click the cube to which the cube roles apply, and then click **Manage Roles**.
3. In Cube Role Manager, limit the list to roles that contain the end user's user name:
   a. Next to **Show**, in the first box, select **Roles containing users**.
   b. In the next box, type the user name.
   c. Click the magnifying glass button.
4. Select the cube roles that contain the end user:
   a. Click the first cube role in the list.
   b. While you hold down SHIFT, click the last cube role in the list.
5. Click **Test Role**. Cube Browser is displayed to simulate the browsing

experience of the end user. If the combination of the custom rules cannot be resolved, an error message is displayed.

6. In Cube Browser, drag restricted dimensions to the data viewing pane and drill down to identify the allowed and denied levels and members.

Cell Security

In a cube role, you can implement cell security to limit the cube cells that end users in the role can view as they browse cubes. You can also grant read/write access to a write-enabled cube and limit the cells that end users in the role can update. You do this by selecting a policy and by selecting a rule or defining a custom rule for each permission.

Cell security is optional. If you do not specify cell security, end users see all cell values in cubes they are authorized to access. (However, if dimension security is specified, cells for some members might not be viewable.) If a cube is write-enabled, end users cannot update cell values. If one or more of a virtual cube's component cubes are write-enabled, end users cannot update the cell values of virtual cubes.

If a policy or rule permits updates to a cell, it can be updated if it is an atomic cell. If the cell is not atomic, it can be updated only if the client application provides a way of dispersing the update over the subordinate atomic cells. For example, in a client application a write-enabled cube is displayed with the lowest level of every dimension except Time. On the axis for the Time dimension, the nonatomic cells for months are displayed, but the subordinate atomic cells for days are not. (Days is the lowest level in the Time dimension.) A cell for June can be updated by adding $90 if the client application provides a way of dividing the +$90 update into thirty +$3 updates, one to each of the cells for the 30 days in June. Dispersion methods other than simple division can also be used. The UPDATE CUBE statement provides several methods. For more information, see UPDATE CUBE Statement.

Cell security is defined in the **Cells** tab of the **Cube Role** dialog box.

Policies

When you specify cell security, for each cube role you can select one of the following policies.

Policy	Description

Unrestricted Read	End users can view all cell values. This policy is the default.
Unrestricted Read/Write	End users can view and update all cell values.
Advanced	End users can view and update only the cell values you specify in the permissions and rules for cell security.

You can select the Unrestricted Read/Write policy for a cube only if it has been write-enabled. If you select this policy, and the cube is subsequently write-disabled, this policy is disabled, and end users cannot update the cube's cell values. You can select this policy for a virtual cube only if one or more of its component cubes have been write-enabled. If you select this policy, and all the component cubes are subsequently write-disabled, this policy is disabled, and end users cannot update the virtual cube's cell values.

Permissions and Rules for the Advanced Policy

If you select the Advanced policy, you can define three permissions: read, read contingent, and read/write. Other policies do not involve permissions or rules. The permissions you define for Advanced policy allow further definition through rules. These rules are similar to the rules you set for roles.

Read Permission (Advanced Policy)

This permission determines which cell values are viewable. Cells specified in this permission are viewable regardless of whether they are derived from other cells that are not viewable. For example, the calculate member Profit is derived from cells for measures Sales and Cost. (Profit equals Sales minus Cost.) If Profit is included in the read permission, its cells are viewable even if cells for Sales or Cost are not.

Note Including derived cells in the read permission incurs the risk that end users might determine cell values they cannot view. For example, if cells for Profit and Cost are viewable, but cells for Sales are not, end users can determine Sales values by adding Profit and Cost values.

You can select one of the following rules for the read permission.

Rule	Description

Unrestricted	End users can view all cell values. This rule is the default.
Fully Restricted	End users can view only the cell values specified in the read/write permission or read contingent permission, subject to the limitations of the read contingent permission described later in this topic.
Custom	This rule provides the most flexibility. You can write an expression in Multidimensional Expressions (MDX) to identify the cell values that are viewable and that are not viewable.

Read Contingent Permission (Advanced Policy)

This permission determines which cell values are viewable. However, cells specified in this permission and derived from other cells are viewable only if all the other cells are also viewable (that is, included in the read permission; or included in read contingent permission but not derived). Cells specified in this permission but not derived from other cells are viewable.

For example, if Profit is included in the read contingent permission, its cells are viewable only if cells for both Sales and Cost are included in the read permission, or if they are included in the read contingent permission but not derived from other cells. If Cost was included in the read contingent permission and derived from other cells, Profit would be viewable only if those cells were included in the read permission or if they were included in read contingent permission but not derived. Thus, with the read contingent permission, a chain of contingencies can be created when a cell is derived from others, which in turn are derived from others, and possibly so on.

If a cell is included in both the read and read contingent permissions, the read permission is enforced, but the read contingent permission is not.

Rule	**Description**
Unrestricted	End users can view all cell values that are not derived from other cells. If a cell value is derived from other cells, it is viewable if all the other cells are included

	in the read or read/write permission.
Fully Restricted	End users can view only the cell values specified in the read permission or read/write permission. This rule is the default.
Custom	This rule provides the most flexibility. You can write an MDX expression to identify the cell values that are viewable and that are not viewable, subject to the limitations of the read contingent permission described earlier in this topic.

If the rules define a cell as not viewable, the cell itself is visible but its value is not. Thus, cell security does not restrict the members that are visible and thereby the size of the visible cube. Rather, it can limit the ability to view the cell values associated with specified members. To limit the members that are viewable, use dimension security. For more information, see Dimension Security.

If the rules define a cell as not viewable, and an end user in the cube role queries this cell, by default the Analysis server returns the formatted value #N/A. The end user sees this value for the cell unless the client application translates the value, or another value is specified by setting the **Secured Cell Value** property in the connection string. For more information about this property, see Secured Cell Value Property.

Read/write Permission (Advanced Policy)

This permission determines which cell values are updatable.

You can define and grant this permission for a cube only if it has been write-enabled. If you grant this permission, and the cube is subsequently write-disabled, this permission is disabled and end users cannot update the cube's cell values.

Cells specified in the read/write permission are also viewable as if they were specified in the read permission, not the read contingent permission.

Rule	Description
Unrestricted	End users can update all cell values.

Fully Restricted	End users cannot update cell values.
Custom	This rule provides the most flexibility. You can write an MDX expression to identify the cell values that are updatable and that are not updatable.

See Also

Creating Cube Roles

Custom Rules in Cell Security

Maintaining Write-Enabled Cubes and Writeback Data

Write-Enabled Cubes

Custom Rules in Cell Security

In a cube role, by defining a custom rule for cell security, you can specify which cube cells can be accessed by end users in the role.

Each custom rule contains an expression written in Multidimensional Expressions (MDX) that allows or denies access to specific cells or groups of cells. The MDX expression resolves to either **True** or **False** for each cell (atomic and nonatomic) in the cube. (If the MDX expression resolves to a numeric value, any nonzero value is evaluated **True**, and zero is evaluated **False**.) If the expression resolves to **True** for a cell, access is allowed. If it resolves to **False**, access is denied.

Custom rules are defined in the **Cube Cell Security** dialog box.

A custom rule provides two alternative approaches for each dimension in the cube. You can either specify the members whose cells can be accessed, or you can specify the members whose cells cannot be accessed.

If you can specify the member whose cells can be accessed, cells of other members cannot be accessed. You can use the equality operator (=) in the MDX expression to identify the cells that can be accessed. Example 1 illustrates this concept.

If you specify the members whose cells cannot be accessed, the cells of other members can be accessed. You can use the inequality operator (<>) in the MDX expression to identify the cells that cannot be accessed. Example 2 illustrates this concept.

In the MDX expression, it is not necessary to include every dimension in the cube. Omitted dimensions do not place restrictions on cell access; that is, cells for all their members can be accessed unless denied by way of a dimension in the MDX expression. If you want to restrict access by only one dimension, a relatively simple MDX expression usually suffices. The complexity of the MDX expression depends largely on the number of dimensions it includes.

By allowing access to cells for specific members in some dimensions and denying access to cells for specific members in other dimensions, you can exercise great flexibility in defining cell security. In fact, you can allow or deny access to any possible combination of cells. Following are examples of functions to use in specific scenarios.

To allow or deny access to cells for a specific member or measure, you can use the MDX functions **CurrentMember** and **Name** in combination. Examples 1 and 2 illustrate this concept. If you specify a member name that is not unique within the dimension, use the **UniqueName** function instead of the **Name** function.

To allow or deny access to the cells for a member and its descendants, include the **Ancestor** function in the expression. Example 3 illustrates this concept. To allow or deny access to cells based on criteria in multiple dimensions, create an expression for each dimension and combine them with AND or OR into one expression. Examples 4, 5, and 6 illustrate this concept.

Example Custom Rules in Cell Security

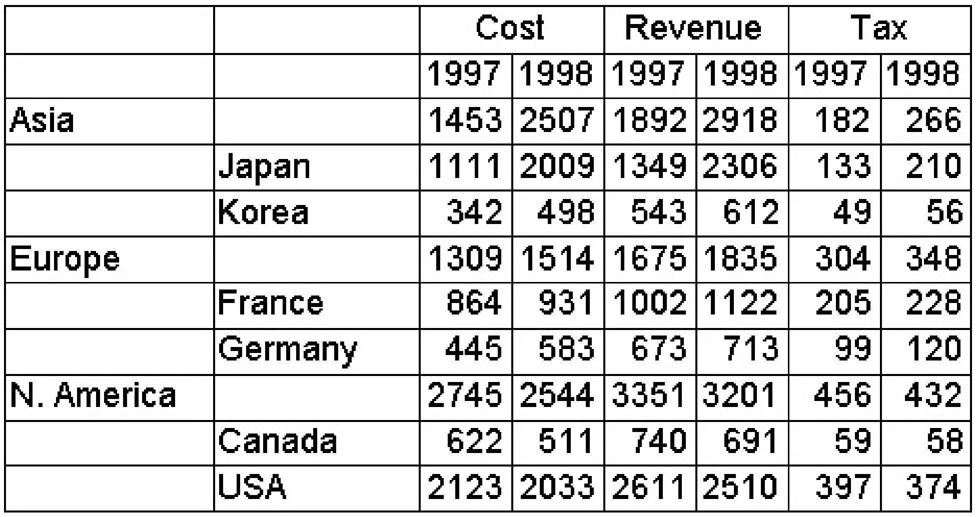
The examples are for a cube defined as follows:

- Measures:
    - Cost
    - Revenue
    - Tax
- Time dimension with levels:
    - Year, limited to members 1997 and 1998
    - Month
- Geography dimension with levels:
    - Continent, limited to members Asia, Europe, and N. America
    - Country, limited to members Japan, Korea, France, Germany,

Canada, and USA

- City

Before cell security is implemented, a dataset returned from this cube looks like the following.



		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia		1453	2507	1892	2918	182	266
	Japan	1111	2009	1349	2306	133	210
	Korea	342	498	543	612	49	56
Europe		1309	1514	1675	1835	304	348
	France	864	931	1002	1122	205	228
	Germany	445	583	673	713	99	120
N. America		2745	2544	3351	3201	456	432
	Canada	622	511	740	691	59	58
	USA	2123	2033	2611	2510	397	374

Each example defines alternative cell security for this cube. Each example includes an expression in Multidimensional Expressions (MDX) in a custom rule for a read permission. The result on the preceding dataset is shown for each example.

The examples assume that the client application translates the formatted value #N/A, which indicates that access to a cell is denied, to a null value.

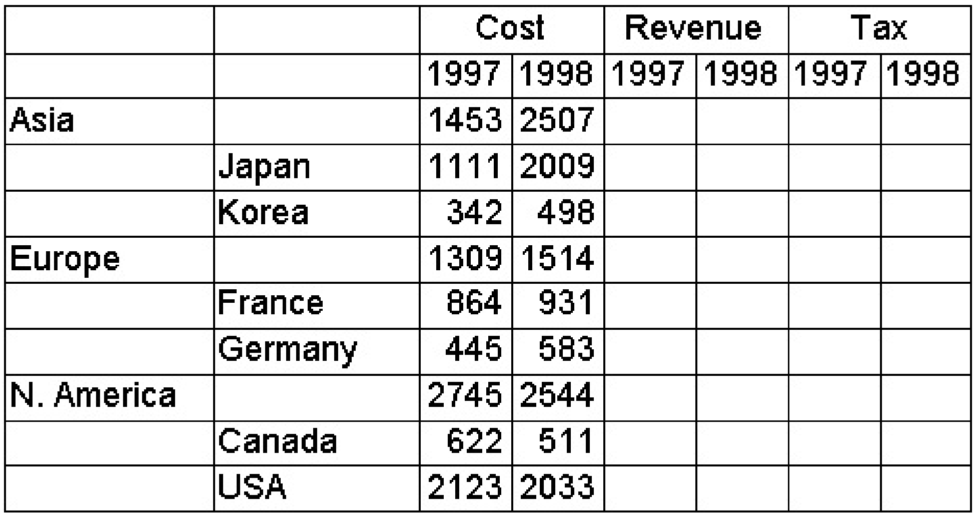
The examples are applied to read permissions so that the effect of the MDX expression on the dataset can be visually demonstrated. However, the examples are also applicable to read/write permissions. That is, if an example's expression was used in a read/write permission, it would allow and deny access to the same cells as in the read permission. The examples would also be applicable to read contingent permissions if none of the displayed cells were derived from other cells.

Example 1

The following MDX expression allows access to cells for the measure Cost, but denies access to cells for all other measures.

Measures.CurrentMember.Name = "Cost"

In a read permission, this expression causes the example dataset to be modified as follows.



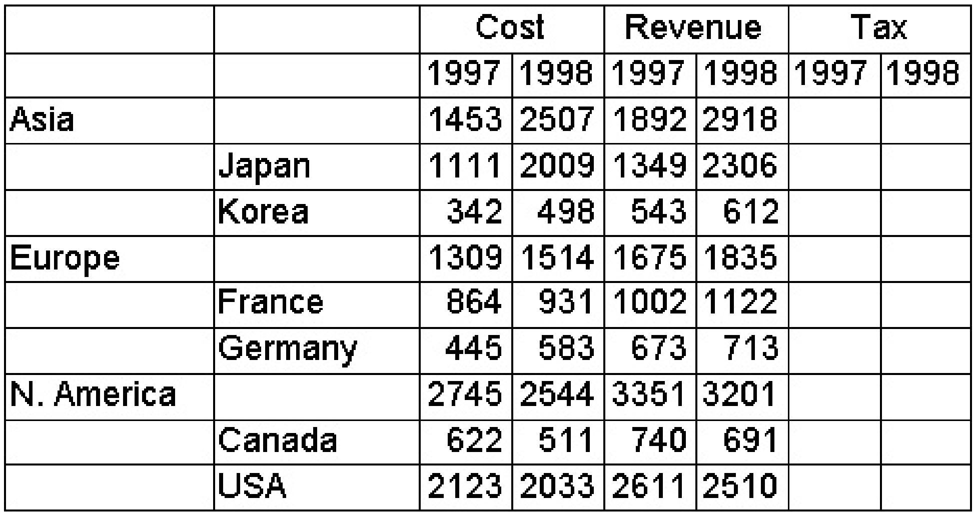
		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia		1453	2507				
	Japan	1111	2009				
	Korea	342	498				
Europe		1309	1514				
	France	864	931				
	Germany	445	583				
N. America		2745	2544				
	Canada	622	511				
	USA	2123	2033				

Example 2

The following MDX expression denies access to cells for the measure Tax but allows access to cells for all other measures.

Measures.CurrentMember.Name <> "Tax"

In a read permission, this expression causes the example dataset to be modified as follows.



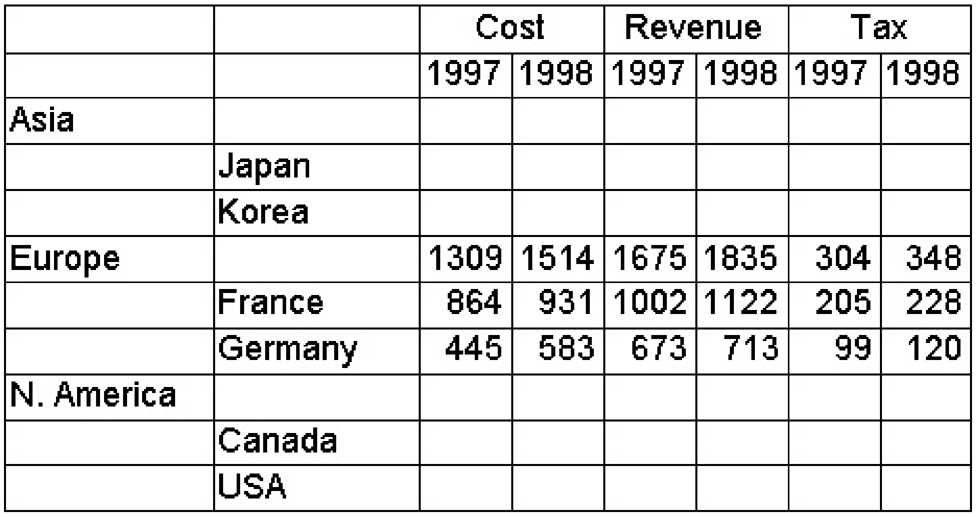
		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia		1453	2507	1892	2918		
	Japan	1111	2009	1349	2306		
	Korea	342	498	543	612		
Europe		1309	1514	1675	1835		
	France	864	931	1002	1122		
	Germany	445	583	673	713		
N. America		2745	2544	3351	3201		
	Canada	622	511	740	691		
	USA	2123	2033	2611	2510		

Example 3

The following MDX expression allows access to cells for the member Europe and its descendants (France, Germany, and their City level members) in the Geography dimension, but denies access to cells for all other members in that dimension.

```
Ancestor(Geography.CurrentMember,[Continent]).Name = "Europe"
```

In a read permission, this expression causes the example dataset to be modified as follows.



		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia							
	Japan						
	Korea						
Europe		1309	1514	1675	1835	304	348
	France	864	931	1002	1122	205	228
	Germany	445	583	673	713	99	120
N. America							
	Canada						
	USA						

Example 4

The following MDX expression allows access to cells for the member N. America and its descendants in the Geography dimension, but denies access to cells for all other members in that dimension. It also allows access to cells for the member 1998 and its descendants in the Time dimension, but denies access to cells for all other members in that dimension.

```
Ancestor(Geography.CurrentMember,[Continent]).Name = "N. Americ
   Ancestor(Time.CurrentMember,[Year]).Name = "1998"
```

In a read permission, this expression causes the example dataset to be modified as follows.

		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia							
	Japan						
	Korea						
Europe							
	France						
	Germany						
N. America			2544		3201		432
	Canada		511		691		58
	USA		2033		2510		374

Example 5

The following MDX expression allows access to cells for the member Asia and its descendants in the Geography dimension, but denies access to cells for all other members in that dimension. It also denies access to cells for the measure Revenue, including Revenue cells for Asia and its descendants.

```
Ancestor(Geography.CurrentMember,[Continent]).Name = "Asia" AN
   Measures.CurrentMember.Name <> "Revenue"
```

In a read permission, this expression causes the example dataset to be modified as follows.

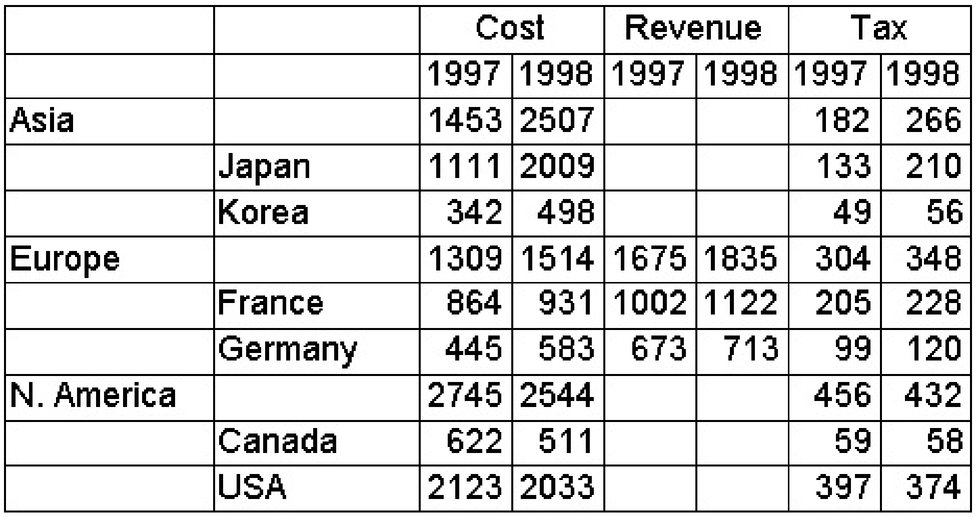
		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia		1453	2507			182	266
	Japan	1111	2009			133	210
	Korea	342	498			49	56
Europe							
	France						
	Germany						
N. America							
	Canada						
	USA						

Example 6

The following MDX expression denies access to cells for the measure Revenue except for those Revenue cells that are also for Europe or one of its descendants in the Geography dimension.

```
Ancestor(Geography.CurrentMember, [Continent]).Name = "Europe" (
   Measures.CurrentMember.Name <> "Revenue"
```

In a read permission, this expression causes the example dataset to be modified as follows.



		Cost		Revenue		Tax	
		1997	1998	1997	1998	1997	1998
Asia		1453	2507			182	266
	Japan	1111	2009			133	210
	Korea	342	498			49	56
Europe		1309	1514	1675	1835	304	348
	France	864	931	1002	1122	205	228
	Germany	445	583	673	713	99	120
N. America		2745	2544			456	432
	Canada	622	511			59	58
	USA	2123	2033			397	374

See Also

Ancestor

CurrentMember

Name

UniqueName

Mining Model Security

The type and scope of access to a data mining model by end users in a mining model role is determined by the settings in the mining model role. An end user can access only those mining models that are assigned a role containing that end user's user name.

A database role provides defaults for the mining model roles of the same name, but default role memberships can be overridden in the mining model roles. After a database role is created, it can be assigned to any mining model in the database. This action grants the end users in the database role access to the mining model and creates a mining model role with the same name as the database role. Database roles are assigned to cubes in the **Mining Models** tab of the **Database Role** dialog box or in Mining Model Role Manager.

See Also

Creating Mining Model Roles

Creating Security Roles

Database, Cube, and Mining Model Roles

Database Role Manager

Mining Model Role Manager

Analysis Services

Operational Considerations

This topic describes operational considerations for security in Microsoft® SQL Server™ 2000 Analysis Services. These considerations are related to Analysis server administration.

Service Logon Account Permissions to Data Sources

The service name for Analysis Services is MSSQLServerOLAPService. If Microsoft Windows NT® 4.0 or Windows® 2000 integrated security is used, the logon account associated with this service must have permissions to access data sources that Analysis Services administrators can access through Analysis Manager. Otherwise, Analysis Services administrators will not be able to process the objects they maintain using Analysis Manager.

To maintain the logon account, use the Services application in Control Panel.

Accessing Your Cube from Another Workstation

An administrator who creates a cube can be denied access to the cube. This can occur when the administrator logs on to a workstation other than the one hosting Analysis Services and attempts to view data in the cube on the server computer. A common cause for this problem is that the administrator was logged on under a local account on the server computer when the cube was created, and then logged on under a local account on the second computer. The cube owner's access control list (ACL) reflects the local account on the server computer, not the local account on the second computer, and the administrator is denied access.

To avoid this problem, you have two options:

- Always log on as a domain account when you create cubes, and then log on as the same domain account on other computers.

- Assign a role to the cube after it is created. You are then able to access the cube from other computers, if you log on as an account granted access by the role.

Lapse Between Change to End User's Access and Effect of Change

The time that elapses between a change to an end user's access defined in an Analysis Services role and the actual effect of the change depends on the value of the **Auto Synch Period** initialization property, the end user's actions, and how long the end user maintains a connection. The value of this property controls the frequency (in milliseconds) of client/server synchronization, including revalidation of end users' access. This value defaults to 10,000 milliseconds (10 seconds), but is passed to Analysis Services in each connection string. Thus, the default can be overridden by end users and client applications and can vary from end user to end user and client application to client application.

If the **Auto Synch Period** property is set to null or 0 (zero), synchronization does not occur at a constant interval. It occurs due to end users' actions; therefore, the time that synchronization will occur cannot be predicted accurately. In this case, changes made to an end user's access while the end user is connected to a cube do not take effect until synchronization occurs or the end user disconnects from the cube. After an end user has been granted access to a cube, that end user can remain connected to the cube for the duration of a query session until synchronization occurs. An end user cannot be forcibly disconnected from a cube during a query session after access has been granted. If the end user's access is removed during the query session, the end user will not be able to reconnect to the cube after disconnecting from it.

If the **Auto Synch Period** property is set to a nonnull, nonzero value, at the specified interval, end users' logon user names and authorizations are compared to their access defined in Analysis Services roles. At that time, changes to an end user's access that occurred since the last synchronization take effect immediately. For example, if an end user's access to a cube has been removed, the end user is immediately unable to access the cube.

For more information about the **Auto Synch Period** property, see Auto Synch Period Property.

Protecting Data

It is important that you protect the security of your data. As with all database products, this includes judicious assignment of administrative access. All users who have administrative access to Analysis servers should be careful when they use Web browsers, productivity applications, and e-mail.

It is recommended that you establish specific Windows NT 4.0 or Windows 2000 user accounts to administer Analysis Services and require administrators to refrain from accessing Web pages, productivity applications, and e-mail applications that support scripts or macros when using these administrative accounts. If it is necessary to use an application that supports scripts or macros when you are logged on as an administrator, set security to the highest level and never accept any control or object that is not marked script safe. Decision Support Objects (DSO) is not marked script safe, and your browser will provide a prompt before loading DSO. You should reject the loading of DSO in this way unless you are certain the application loading it is trusted.

It is also recommended that you use Windows NT 4.0 or Windows 2000 integrated security for connections between an Analysis server and SQL Server used as a data source.

See Also

User Accounts and Groups

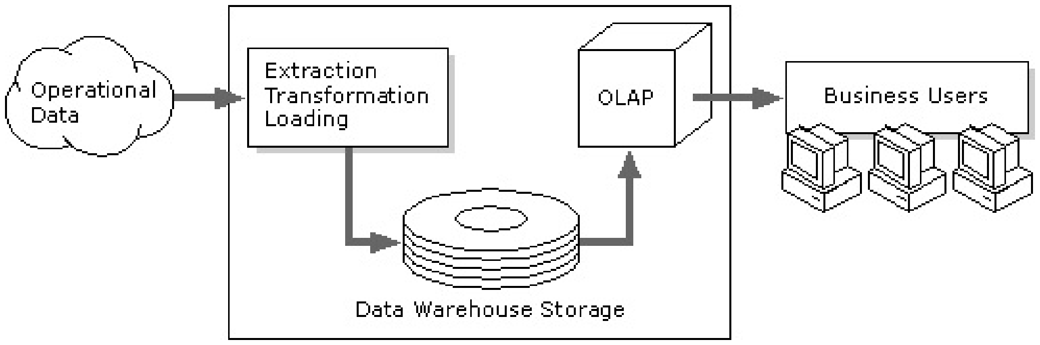
Analysis Services

Data Warehousing and OLAP

Although sometimes used interchangeably, the terms *data warehousing* and *online analytical processing* (OLAP) apply to different components of systems often referred to as decision support systems or business intelligence systems. Components of these types of systems include databases and applications that provide the tools analysts need to support organizational decision-making.

A data warehouse is a database containing data that usually represents the business history of an organization. This historical data is used for analysis that supports business decisions at many levels, from strategic planning to performance evaluation of a discrete organizational unit. Data in a data warehouse is organized to support analysis rather than to process real-time transactions as in online transaction processing systems (OLTP).

OLAP technology enables data warehouses to be used effectively for online analysis, providing rapid responses to iterative complex analytical queries. OLAP's multidimensional data model and data aggregation techniques organize and summarize large amounts of data so it can be evaluated quickly using online analysis and graphical tools. The answer to a query into historical data often leads to subsequent queries as the analyst searches for answers or explores possibilities. OLAP systems provide the speed and flexibility to support the analyst in real time.



See Also

Creating and Using Data Warehouses Overview

Analysis Services

About Data Warehouses

A data warehouse is often used as the basis for a decision support system. Data warehouses are designed to overcome problems encountered when an organization attempts to perform strategic analysis using the same database that is used for online transaction processing (OLTP).

OLTP systems typically:

- Support large numbers of concurrent users who are actively adding and modifying data.
- Represent the constantly changing state of an organization but don't save its history.
- Contain large amounts of data, including extensive data used to verify transactions.
- Have complex structures.
- Are tuned to be responsive to transaction activity.
- Provide the technology infrastructure to support the day-to-day operations of an organization.

Difficulties often encountered when OLTP databases are used for online analysis include the following:

- Analysts do not have the technical expertise required to create ad hoc queries against the complex data structure.
- Analytical queries that summarize large volumes of data adversely affect the ability of the system to respond to online transactions.

- System performance when responding to complex analysis queries can be slow or unpredictable, providing inadequate support to online analytical users.
- Constantly changing data interferes with the consistency of analytical information.
- Security becomes more complicated when online analysis is combined with online transaction processing.

Data warehousing provides one of the keys to solving these problems, by organizing data for the purpose of analysis. Data warehouses:

- Can combine data from heterogeneous data sources into a single homogenous structure.
- Organize data in simplified structures for efficiency of analytical queries rather than for transaction processing.
- Contain transformed data that is valid, consistent, consolidated, and formatted for analysis.
- Provide stable data that represents business history.
- Are updated periodically with additional data rather than frequent transactions.
- Simplify security requirements.
- Provide a database organized for OLAP rather than OLTP.

A data mart is a special form of data warehouse, typically containing a topic-oriented subset of enterprise data appropriate to a specific business function.

Microsoft® SQL Server™ 2000 provides many essential tools for building data warehouses and data marts, including Data Transformation Services (DTS).

Analysis Services

About OLAP

Whereas data warehouses and data marts are the data stores for analysis data, online analytical processing (OLAP) is the technology that enables client applications to efficiently access this data. OLAP provides many benefits to analytical users, for example:

- An intuitive multidimensional data model makes it easy to select, navigate, and explore the data.

- An analytical query language provides power to explore complex business data relationships.

- Precalculation of frequently queried data enables very fast response time to ad hoc queries.

Microsoft® SQL Server™ 2000 Analysis Services is a robust OLAP tool that can be used with data stored in various data warehouse databases, including SQL Server, Microsoft Access, and Oracle databases. For more information, see Analysis Services Features.

Analysis Services

OLAP and Data Warehouses

OLAP provides a multidimensional presentation of data warehouse data, creating cubes that organize and summarize data for efficient analytical querying. The design of the data warehouse structure can affect how easily these cubes can be designed and constructed.

Microsoft® SQL Server™ 2000 Analysis Services relies on the data provided by the data warehouse to be accurate, stable, and to have referential integrity. When creating a data warehouse for use with Analysis Services, these design factors should be considered:

- Use a star schema if possible.

  If a snowflake schema is needed, minimize the number of dimension tables beyond the first level from the fact table.

- Design dimension tables for the users.

  Dimension tables should include meaningful information about the facts that users will want to explore, such as the color or size of a product.

- Apply commonsense normalization to dimension table design.

  Unrelated data should not be combined into a single dimension table, and data should not be repeated in multiple dimension tables. For example, create a separate customer dimension instead of repeating customer information in more than one dimension table.

- Do not over-summarize in the fact table.

  Retain the finest level of granularity users need to access, and keep all fact table records at the same level of detail. Analysis Services is designed to create and manage summary data from highly granular data warehouses without penalizing users in query response time.

- Use a common fact table structure for similar data.

  Data intended to be used in the same cube can be stored in multiple fact tables, but those tables must have the same structure.

- Do not create auxiliary tables of summarized data.

  Analysis Services precalculates summaries into structures that are designed for query efficiency. Other auxiliary summarization tables are not used.

- Create indexes on key fields.

  For each dimension table, create an index on its key column. For each fact table, create a single index on the combination of columns that contain the foreign keys of the dimension tables associated with the fact table. Analysis Services uses these indexes when it loads multidimensional data structures and calculates summary data. These indexes significantly improve cube processing performance.

- Ensure referential integrity.

  It is important that all facts be represented in all dimension tables. Facts in a fact table that do not have a corresponding key in a dimension table can cause errors or fact table rows to be ignored if the fact and dimension tables are used in the same cube.

- Design a data update strategy.

  When data is added to or changed in the data warehouse, cubes that have been built from previous data must be updated before the new data is available to users. Incorporating additional data into cubes requires less time than rebuilding cubes when existing data changes. For more information, see Maintaining OLAP Data.

See Also

Creating and Using Data Warehouses Overview

Analysis Services

Maintaining OLAP Data

The purpose of Microsoft® SQL Server™ 2000 Analysis Services is to provide rapid analytical access to data warehouse data. To accomplish this purpose, Analysis Services creates multidimensional cubes from data in the data warehouse fact and dimension tables. Numerical measures are also summarized into preaggregated values during cube construction. Cubes are stored in multidimensional structures that are designed for rapid query response, combining preaggregated information with raw fact data to respond to a wide variety of queries.

Cubes can contain data summarized, copied, or read directly from the data warehouse. Changes to the structure of the data warehouse or the data contained in it can affect the integrity and accuracy of cubes that have been created from the data warehouse. Because Analysis Services provides continuous online access to cubes, changes to the underlying data warehouse must be approached with a clear understanding of their effects on cubes and how to manage the synchronization of data in the data warehouse with data in cubes.

OLAP data must be updated after data warehouse data is changed. You process OLAP cubes, dimensions, and partitions to incorporate new or changed data from the data warehouse. The method of processing an OLAP object depends on the object and type of change made to the data warehouse, such as data addition, data change, or structural change.

Real-time OLAP is a feature that uses real-time cubes to automatically synchronize cube data with changes in the underlying relational database. Real-time cubes can be used for applications that need to monitor and analyze live data, and are intended to extend OLAP capabilities rather than replace traditional cube designs and applications.

Changes in the Data Warehouse

Data is usually added periodically to the data warehouse to include more recent information about the organization's business activities. Changes to data already in the data warehouse are less frequent and usually made only to incorporate corrections to errors discovered in the source from which the data was extracted, or to restructure data due to organizational changes. Structural changes to the data warehouse design typically are the least common.

Data Additions

It is common to add new data to the data warehouse. Cube information available online to client applications can be affected when data is added to the data warehouse due to interaction between the data and cube partitions. You can manage the effects of adding data to the data warehouse by carefully defining partition filters, and by designing a strategy to synchronize OLAP and data warehouse data.

Data Changes

Changes to correct errors in a data warehouse can be minimized by applying care during the data transformation, validation, and scrubbing operations. Other changes to existing data warehouse data can arise from changes in the structure of an organization or its products. For example, reorganizing products into different categories can require significant changes to data in the data warehouse, as well as to reports derived from the data warehouse. In some cases, such changes can require the complete redesign of cubes. In other cases, the redesign of dimensions and the processing of all cubes that use those dimensions may be all that is required.

Changes to correct errors in basic data should be incorporated in the source database, usually the OLTP business database, and then migrated to the data warehouse in a controlled manner. Many business OLTP database designs require changes to be made by a transaction that offsets the incorrect data and applies new correct data. It is often easier to manage the impact of such correction transactions on OLAP data. Cubes can incorporate new data transactions that correct value errors, such as an incorrect sale value. However, transactions that move a fact from one dimension member to another, such as a sale posted to the wrong customer, can affect the results of aggregate functions such as **Avg**. This is true for non-OLAP databases as well; if an original sale order is zeroed out but the record remains in the database, it will be included in the count of sales records and affect the calculation.

Depending on cube storage design, changes to data in the fact table can affect the accuracy of queries to a cube until the cube is processed. The **Refresh data** processing option can be used to reload the cube's data and recalculate the aggregations. Because aggregation design remains the same, the **Refresh data** processing option is faster than the complete **Full process** processing option.

Dimension hierarchies can be affected by changes to data in the data warehouse dimension tables even though the table schema remains the same. The dimension hierarchy is based on relationships between members in a dimension table. When these relationships are changed (for example, when cities are reorganized into different sales regions), the dimension structure must be rebuilt.

Referential integrity must be maintained when data warehouse data is added, changed, or deleted. Loss of referential integrity can result in errors during cube processing, fact table records being bypassed, or inaccurate OLAP information.

Structure Changes

The structure of OLAP cubes and dimensions can be affected by changes to the design of the data warehouse such as the addition, deletion, or alteration of tables, or relationships between tables. When the structure changes, you must modify the design of affected cubes and dimensions, redefine partitions and aggregations, and completely process the modified cubes and dimensions.

Synchronizing OLAP and Data Warehouse Data

Valid cubes are online and available to client applications at all times when the Analysis server is running. Because of the potential for interaction of OLAP cube partitions with data in the data warehouse, the design of the data warehouse should include a synchronization strategy to enable the addition of data without causing cubes to provide incorrect answers to queries in cubes available to online client applications.

One strategy for managing additions to data warehouse and OLAP data is to design a batch update system. In this strategy, all data in the data warehouse fact table includes a batch number in each record. When you design a cube, add an expression to the filter for each of the cube's partitions to specify the largest batch number applicable, for example, "... AND DWBatch <= 33 ..." When additions to the fact table need to be made, include a new, higher batch number in the new records. Cubes are unaffected by these added records because the cube partitions are restricted to reading data from previous batches only.

Data added to a dimension table does not affect existing cube private or shared dimensions until the dimensions are processed. A batch number in dimension table records is not necessary, but it can be useful in ensuring continued referential integrity.

Dimensions and cubes or partitions can be processed to incorporate new data after a batch of data has been added to the fact table and dimension tables. Shared dimensions should be processed before the cubes that use them. To add new members to a dimension that do not affect the dimension's structure, use the **Incremental update** option. To add new members and rebuild the dimension's structure, use the **Rebuild the dimension structure** option. Note that when a shared dimension is processed with the **Rebuild the dimension structure** option, all cubes that incorporate that dimension will immediately become unavailable to client applications and must be processed before they can be used again. However, when a shared dimension is processed using the **Incremental update** option, a cube that uses the shared dimension will display the new members, but the cells associated with those members will remain empty until the cube is updated with new data from the fact table that relates to the new members.

To incorporate a new data batch in a cube, update the filter expression in each of the cube's partitions to include the new batch number, and then process or incrementally update the cube. If a cube's data is divided among multiple partitions, you can use one of the partitions to accumulate new data batches and process that partition only. The cube's other partitions must have filters that exclude new data so that data will be added only to the accumulation partition.

Visibility to Client Applications

When a cube that is currently online is processed by any one of the three processing options (**Full process**, **Incremental update**, or **Refresh data**), the cube remains online until the processing has been completed, at which time the online cube is replaced by the new cube version. When a cube is processed using the **Full process** option, online client applications will be disconnected from the cube when the switch is made to the new version of the cube, and the client applications must individually reconnect to access the new version. When a cube is processed using either the **Incremental update** or the **Refresh data** option, online client applications will not be disconnected from the cube when the processing completes. The new version of the cube will be immediately visible with no break in service.

The processing of a shared dimension can affect cubes that incorporate the dimension in their design. If a shared dimension is processed using the **Rebuild the dimension structure** option, all cubes that use the dimension will immediately become unavailable to client applications and must be processed before they can be used again. If a shared dimension is processed using the **Incremental update** option, cubes that use the dimension remain available to client applications and any new members added to the dimension automatically become available to client applications when the dimension processing is complete. Any such new members will not have fact data associated with them until the cube is updated with new related facts.

See Also

Building and Processing Cubes

Updating Cubes and Dimensions

Analysis Services

Administering Analysis Services

The following topics contain conceptual information about administering Microsoft® SQL Server™ 2000 Analysis Services.

Topic	Description
Before Administering Analysis Services	Describes preliminary steps you must take before administering Analysis Services, such as preparing the data warehouse and defining data sources.
Administrative Tools	Describes the user interface tools you use to work with Analysis Services, including Analysis Manager, Data Transformation Services (DTS), Multidimensional Expressions (MDX), and the **msmdarch** command.
Administrative Tasks	Summarizes necessary administrative tasks and processes.

Analysis Services

Before Administering Analysis Services

Before you begin administering Microsoft® SQL Server™ 2000 Analysis Services, prepare the data warehouse from which you will create cubes or data mining models. Also, if you are planning to specify data sources that are accessed through ODBC drivers, create system data source names (DSNs) for these databases.

Preparing the Data Warehouse

Several objects in Analysis Services are created from data in a data warehouse: cubes, partitions, dimensions, and data mining models. The data warehouse from which an object is created is specified in the data source of the object. For more information about data sources, see Data Sources.

Before you create these objects, prepare the data warehouse so that the objects can be efficiently created and maintained. For more information about the specific actions required to prepare the data warehouse, see OLAP and Data Warehouses.

Defining Data Sources in the ODBC Data Source Administrator

Analysis Services supports OLE DB and ODBC data sources. If you are using only OLE DB data sources as input to Analysis Services, you do not need to use the ODBC Data Source Administrator.

If you are planning to use the Microsoft OLE DB Provider for ODBC drivers for connections to ODBC data sources, before you start Analysis Manager, ensure that all of these ODBC data sources are defined as system DSNs in the ODBC Data Source Administrator.

Note Remote administrators of Analysis Services must also define the same data sources on their computers.

See Also

Specifying Data Sources

Analysis Services

Administrative Tools

Microsoft® SQL Server™ 2000 Analysis Services provides several administrative tools, which are summarized in the following topics.

Topic	Description
Analysis Manager	Describes the user interface for administering Analysis Services.
Using Active Directory with Analysis Services	Details the registration of Analysis servers with Active Directory™.
Using Data Transformation Services with Analysis Services	Summarizes the Analysis Services Processing task, which is used to process cubes, data mining models, and other objects.
Using MDX with Analysis Services	Describes uses, operations, and functions of Multidimensional Expressions (MDX), a syntax for manipulating multidimensional data.
Msmdarch Command	Describes the command used for archiving and restoring Analysis Services databases.

Analysis Services

Analysis Manager

Microsoft® SQL Server™ 2000 Analysis Services includes Analysis Manager, a console application that provides a user interface for accessing Analysis servers and their meta data repositories. Use Analysis Manager to:

- Administer Analysis servers. Multiple users can administer an Analysis server using Analysis Manager. Locking is applied only to the objects being edited and their dependent objects.
- Create databases and specify data sources.
- Build and process cubes.
- Create and process data mining models.
- Specify storage options and optimize query performance.
- Manage security.
- Browse data sources, shared dimensions, security roles, and other objects.
- Find links for third-party client applications, support resources, Help updates, and product news.
- Work through the Analysis Manager tutorial. This step-by-step tutorial guides you through building, deploying, managing, and enhancing an OLAP cube. This tutorial also provides instruction on how to create a data mining model.

- Obtain information about the complete SQL Server 2000 product.

In addition to the tutorial, the HTML (that is, right) pane of Analysis Manager assists you in learning the concepts and terminology of online analytical processing (OLAP) and data mining, as well as how to store and manage the associated data.

MMC

Microsoft® Management Console (MMC) is a common console framework for server and network management applications known as snap-ins. Snap-ins allow administrators to more effectively manage network resources. Analysis Manager is a snap-in to MMC.

Starting Analysis Manager

To start Analysis Manager

- On your desktop click **Start**, point to **Programs**, point to **Microsoft SQL Server**, point to **Analysis Services**, and then click **Analysis Manager**.

The Analysis Manager Tree Pane

Analysis Manager is represented as a folder named Analysis Servers located beneath the Console Root folder. This topic describes the Analysis Manager hierarchy and the location of its components, which make up the Analysis Manager tree pane.

Analysis Servers

The Analysis Servers folder contains an icon for each Analysis server registered in Analysis Manager. The name beside the icon is the same as the server name.

Databases

Each Analysis server contains one or more databases. Each database is represented by an icon beneath the Analysis server icon.

Beneath each database icon is a:

- Data Sources folder for the data sources specified in the database.
- Cubes folder for the cubes in the database.
- Shared Dimensions folder for the shared dimensions in the database. These dimensions are defined at the database level and can be shared among the cubes in the database.
- Mining Models folder for data mining models stored in the database.
- Database Roles icon that represents all of the database roles in the database.

Data Sources

Beneath each database icon is a Data Sources folder for the data sources specified in the database. A data source maintains OLE DB provider information, server connection information, network settings, connection time-out, and access permissions. A database can contain multiple data sources in its Data Sources folder.

Cubes

Beneath each database icon is a Cubes folder for the cubes in the database. Each cube is represented by an icon. Three varieties of cubes are depicted in the Analysis Manager tree pane: regular, linked, and virtual.

Regular Cubes

In a Cubes folder, each regular cube is represented by an icon. Beneath each regular cube icon is a:

- Partitions folder that contains an icon for each partition in the cube.

- Cube Roles icon that represents all of the cube roles for the cube.

To see the dimensions, measures, and other components in a regular cube, right-click its icon and then click **Edit**.

Linked Cubes

In a Cubes folder, each linked cube is represented by an icon. Beneath each linked cube icon is a Cube Roles icon that represents all of the cube roles for the cube.

Virtual Cubes

In a Cubes folder, each virtual cube is represented by an icon. Beneath each virtual cube icon is a Cube Roles icon that represents all of the cube roles for the cube.

To see the dimensions, measures, and other components in a virtual cube, right-click its icon and then click **Edit**.

Partitions

A cube's Partitions folder contains an icon for each partition in the cube. There are two types of partitions depicted in the Analysis Manager tree pane: local and remote.

Local Partitions

In a Partitions folder, each local partition is represented by an icon. To access the settings for a partition, right-click its icon, and then click **Edit**.

Remote Partitions

In a Partitions folder, each remote partition is represented by an icon. To access the settings for a partition, right-click its icon, and then click **Edit**.

Cube Roles

Beneath a cube, a single Cube Roles icon represents all of the cube roles for the cube. To access the roles, right-click the icon, and then click **Manage Roles**.

Shared Dimensions

Beneath each database icon is a Shared Dimensions folder that contains an icon for each shared dimension in the database. These dimensions can be included in any cube in the database. Four varieties of shared dimensions are depicted in the Analysis Manager tree pane: regular, virtual, parent-child, and data mining.

Regular Dimensions

In a Shared Dimensions folder, each regular dimension is represented by an icon.

To see the levels, members, and other components in a dimension, right-click its icon, and then click **Edit**.

Virtual Dimensions

In a Shared Dimensions folder, each virtual dimension is represented by an icon.

To see the levels, members, and other components in a dimension, right-click its icon, and then click **Edit**.

Parent-Child Dimensions

In a Shared Dimensions folder, each parent-child dimension is represented by an icon.

To see the levels, members, and other components in a dimension, right-click its icon, and then click **Edit**.

Data Mining Dimensions

In a Shared Dimensions folder, each data mining dimension is represented by an icon.

To see the levels, members, and other components in a dimension, right-click its icon, and then click **Edit**.

Mining Models

Beneath each database icon is a Mining Models folder that contains an icon for each mining model in the database, There are two types of mining models depicted in the Analysis Manager, relational and OLAP.

Relational Mining Models

In a Mining Models folder, each relational mining model is represented by an icon.

To view or modify the structure of a mining model, right-click its icon, and then click **Edit**. To view the content of a mining model, right-click its icon, and then click **Browse**.

OLAP Mining Models

In a Mining Models folder, each OLAP mining model is represented by an icon.

To view or modify the structure of a mining model, right-click its icon, and then click **Edit**. To view the content of a mining model, right-click its icon, and then click **Browse**.

Mining Model Roles

Beneath a mining model, a single Mining Model Roles icon represents all of the mining model roles for the mining model. To access the roles, right-click the icon, and then click **Manage Roles**.

Database Roles

Beneath a database, a single Database Roles icon represents all of the database roles in the database. These roles can be assigned to any cube in the database. To access the roles, right-click the icon, and then click **Manage Roles**.

See Also

Object Architecture

Analysis Services

Using Active Directory with Analysis Services

Microsoft® SQL Server™ 2000 Analysis Services supports Active Directory™. You can register an Analysis server with Active Directory to provide users with an easy way to search for registered servers using the Microsoft Windows® 2000 Search functionality.

For each Analysis server that you register, you can expose server information that the user subsequently uses to select the server. For example, you can provide relevant details, such as the location of the server, the name of the person who maintains the server, how the server is used, the kind of data it stores, and so on. In an enterprise that follows strict naming conventions, the ability to use additional criteria to locate a server can help circumvent server names that are not intuitive or are difficult to remember.

In Analysis Services, support for Active Directory is available for Analysis servers. Specific databases and cubes cannot be registered with Active Directory.

Registering an Analysis Server with Active Directory

During installation, Active Directory is configured to accept registration entries of SQL Server and Analysis Services instances. After installation is complete, you can register an Analysis server to make it available as a shared resource on your intranet. After you register an Analysis server, Windows 2000 notifies Active Directory of the new entry.

Registering an Analysis server with Active Directory is not a substitute for creating user accounts or setting permissions. Users who select a registered server are subject to the security measures that you have set for specific servers, databases, and cubes.

To register an Analysis server, use the **Properties** dialog box. You can access the **Properties** dialog box by right-clicking a server name in the Analysis Manager tree pane, and then clicking **Properties**. Use the **Active Directory** tab to register the server and set additional properties.

The properties that you can set correspond to search criteria that the user defines. When specifying properties, provide values that users are most likely to find helpful. For more information, see Active Directory Tab (Properties Dialog Box).

Searching for an Analysis server using Active Directory requires code that you provide. To support Active Directory searches, you can create a simple application or tool that allows users to find Analysis servers through name-based or keyword searches. Creating an Active Directory application requires the Active Directory Service Interfaces (ADSI). You can use the functions and properties of Active Directory objects to support browsing. For more information about ADSI, go to the MSDN® Web page at the Microsoft Web site and search for ADSI.

Analysis Services

Using Data Transformation Services with Analysis Services

You can use Data Transformation Services (DTS) in Microsoft® SQL Server™ 2000 to process cubes, data mining models, and other objects, and to create prediction tasks based on mining models. For processing activities, a DTS task called the Analysis Services Processing task is provided, and for mining model predictions, the Data Mining Prediction Query task is provided. You can access both tasks within DTS Designer, which provides a user interface for defining DTS packages and tasks. After you define a package that contains an Analysis Services Processing task or a Data Mining Prediction Query task, you can schedule it to execute automatically. For more information, see Processing Objects Using Data Transformation Services.

See Also

DTS Basics

Analysis Services

Using MDX with Analysis Services

Multidimensional Expressions (MDX) is used in a variety of administrative tasks. You can use MDX to create the following items.

Item	Description	For more information, see
Action	An end user-initiated operation upon a selected cube or portion of a cube.	Actions
Calculated member	A dimension member whose value is calculated at run time using an expression that you specify when you define the calculated member.	Calculated Members
Calculated cells	An expression that determines the cube cell values associated with a specific selection of cells. The expression overrides the aggregate functions of measures.	Calculated Cells
Named set	A set of dimension members or a set expression that is created for reuse, for example, in MDX queries.	Named Sets
Custom rollup formula	An expression that determines the cube cell values associated with the members of a dimension level. The expression overrides the aggregate functions of measures.	Custom Rollup Formulas and Custom Member Formulas
Custom member formula	An expression that determines the cube cell values associated with a member. The expression overrides the aggregate functions of measures.	Custom Rollup Formulas and Custom Member Formulas
Custom rule in dimension security	A rule that specifies the dimension levels and members that can be	Custom Rules in Dimension Security

	accessed by a role.	
Custom rule in cell security	A rule that specifies the cube cells that can be accessed by a role.	Custom Rules in Cell Security

For more information about MDX, see MDX.

Analysis Services

Msmdarch Command

You can use the **msmdarch** command to archive and restore databases in Microsoft® SQL Server™ 2000 Analysis Services. You can execute the command in the Command Prompt window or from a .bat file. The **msmdarch** command returns an exit code of 1 if it fails. For more information, see Archiving and Restoring Databases.

To archive an Analysis Services database using the msmdarch command

Analysis Services

Administrative Tasks

These topics contain information you should read before performing administrative tasks or procedures in Microsoft® SQL Server™ 2000 Analysis Services for the first time. Where applicable, topics contain links to step-by-step procedures. For more information about specific objects in Analysis Services and how they work together, see Analysis Services Architecture.

Topic	Description
Configuring Analysis Servers	Contains information about registering servers, setting server properties, migrating repositories, and configuring servers for access from the Internet.
Creating Prerequisite Objects for Cubes	Contains information about creating databases, specifying data sources, and creating dimensions.
Building and Processing Cubes	Describes different methods of building cubes and ways of browsing data before and after processing the cubes for use.
Creating Security Roles	Provides background information about creating roles and rules for different levels of security.
Managing Partitions	Contains information about creating and merging partitions.
Enhancing Dimensions with Optional Features	Describes different ways of including additional information and creating customized views of dimension data, including member properties, custom member formulas, and member groups.
Enhancing Cubes with Optional Features	Describes working with specialized types of cube data and configurations, including calculated members, named sets, actions, write-enabled cubes, and drillthrough.
Updating Cubes and Dimensions	Describes different methods for updating and rebuilding cube and dimension data and structure.
Building and Using Data Mining Models	Contains information about working with mining models.

Archiving, Restoring, and Copying Data	Contains information about archiving and restoring databases and copying and pasting objects in Analysis Manager.
Analyzing and Optimizing Performance	Describes how to use Analysis Manager to produce reports of query history and optimize storage and performance based on that history.
Automating and Scheduling Administrative Tasks	Describes how to use Data Transformation Services (DTS) and batch files to perform certain administrative tasks in Analysis Services.

See Also

Administrative Tools

Before Administering Analysis Services

Analysis Services

Configuring Analysis Servers

The following topics contain conceptual information about configuring Analysis servers in Microsoft® SQL Server™ 2000 Analysis Services.

Topic	Description
Registering Servers	Contains information about what happens when you register an Analysis server and how to register servers.
Setting Server Properties	Describes using the **Properties** dialog box to set server properties.
Migrating Analysis Services Repositories	Describes using the Migrate Repository Wizard to move the Analysis Services repository from one format and location to another.

Registering Servers

The computer on which you install Microsoft® SQL Server™ 2000 Analysis Services is registered automatically as a server. It appears in the Analysis Manager tree pane immediately below the Analysis Servers folder. To access and maintain meta data on other servers, you must register them separately.

Each server stores its own Analysis Services database or databases.

To register an Analysis server

Setting Server Properties

The **Properties** dialog box gives you access to many of the Analysis server settings that are stored in the registry. This dialog box appears when you right-click an Analysis server node in the Analysis Manager tree pane and then click **Properties**.

You can optimize a server through the **Properties** dialog box. This lets you get the best performance possible from Microsoft® SQL Server™ 2000 Analysis Services, while it minimizes the dangers that can occur when settings stored in the registry are manually edited. You can also use the **Properties** dialog box to reset the options in the **Properties** dialog box to their default installation values.

The **Properties** dialog box contains options that control user interfaces, server environment, processing, logging, and add-ins.

See Also

Properties Dialog Box

Migrating Analysis Services Repositories

For each Analysis server, Microsoft® SQL Server™ 2000 Analysis Services creates a repository called the Analysis Services repository to store meta data for the objects of the Analysis server (cubes, dimensions, and so on). By default, this repository is a Microsoft Access (.mdb) database on the server computer where Analysis Services is installed. By default, this database is:

C:\Program Files\Microsoft Analysis Services\Bin\msmdrep.mdb

You can use the Migrate Repository Wizard to migrate this repository to a SQL Server (.mdf) database on the same or another server computer. Later, you can use the wizard to migrate the repository to another SQL Server database.

IMPORTANT After you migrate the repository to a SQL Server database, you cannot migrate the repository back to a Microsoft Access database.

You can change the format of your repository when you migrate it from the default Microsoft Access database to a SQL Server database or from one SQL Server database to another SQL Server database. By default the repository is in SQL Server 7.0 OLAP Services format. You can keep this format when you migrate the repository or you can change the format to use the SQL Server 2000 Meta Data Services repository format. If you choose to use the Meta Data Services format, it is recommended that you store the repository in the **msdb** database, which is where other SQL Server components store their repositories in this format. You can also change the repository from Meta Data Services format to SQL Server 7.0 OLAP Services format when you migrate it from one SQL Server database to another.

Before you start the Migrate Repository Wizard, create or identify the SQL Server database to which you want to migrate the Analysis Services repository.

To start the Migrate Repository Wizard

Analysis Services

Creating Prerequisite Objects for Cubes

Before you can create cubes, you must create or identify supporting objects such as databases and dimensions. These topics provide basic information about these objects and how they work with cubes.

Topic	Description
Creating Databases	Contains information about databases and what they store in Microsoft® SQL Server™ 2000 Analysis Services.
Specifying Data Sources	Contains information about data sources for cubes and the impacts of specifying different data sources after you have built cubes that use them.
Creating Shared Dimensions	Describes the creation of shared dimensions, which are available to multiple cubes.
Creating Virtual Dimensions	Describes the creation of virtual dimensions, which combine existing dimensions.
Viewing Dimension Data	Contains information about how to browse dimension data.

Creating Databases

On each of your Analysis servers, create one or more databases to store your cubes, data mining models, and related objects. Create a database for each group of related cubes and mining models that you plan to create. Each database stores its own cubes, mining models, data sources, shared dimensions, and database roles. Objects that will be shared among multiple cubes and mining models should be stored within the same database.

To create a database, use the **Database** dialog box.

To create a database

Specifying Data Sources

For each of your databases, specify one or more data sources that will provide data for your cubes, data mining models, and other objects. When you specify a data source, Analysis Manager retrieves a list of available OLE DB providers. If you want to use a database accessible through an ODBC driver as a data source, select Microsoft® OLE DB Provider for ODBC Drivers as the OLE DB provider, and then select the ODBC data source name (DSN) for the database as defined in ODBC Data Source Administrator from a separate dropdown list. You can easily select a provider and, if necessary, a DSN from these lists.

Note DSNs defined in ODBC Data Source Administrator must be defined as system DSNs.

If you later edit the data source and change the provider, errors can occur during subsequent processing of objects. For example, in a time dimension, the values for the levels' **Member Key Column** and **Member Name Column** properties are frequently SQL expressions. If the syntax of the expressions is supported by the original provider, but not by the new provider, processing the dimension will fail.

To specify a data source, use the **Data Link Properties** dialog box.

To specify a data source

Creating Shared Dimensions

Dimensions are descriptive categories by which the numeric data (that is, measures) in a cube can be separated for analysis. For example, if the measure of a cube is Cost, and its dimensions are Time, Supplier, and Item Description, users of the cube can separate Cost into various categories of Time, Supplier, and Item Description.

A shared dimension is a dimension available to multiple cubes in a database. You should create shared dimensions for common dimensions, such as Time, that will be added to multiple cubes. You can create a shared dimension by using the Dimension Wizard or Dimension Editor, and you have the option of creating a shared dimension to store results of a data mining analysis when creating an OLAP mining model.

The following topics describe the creation of regular shared dimensions.

Topic	Description
Creating a Shared Dimension with the Wizard	Describes the steps in the Dimension Wizard that create regular shared dimensions.
Creating a Shared Dimension with the Editor	Describes the steps in Dimension Editor you use to create new shared dimensions.
Creating Virtual Dimensions	Provides information about virtual dimensions, which are dimensions based on columns or member properties.
Viewing Dimension Data	Describes how to browse data using Dimension Browser.

See Also

Mining Model Wizard

Creating a Shared Dimension with the Wizard

To create a regular shared dimension quickly and easily, use the Dimension Wizard. The wizard takes you through steps to specify the structure of your shared dimension:

- Required features, such as the dimension's source data
- Features required in some situations, such as dimension levels, dimension table joins, and various types of dimension table columns
- Optional features, such as special properties and custom formulas

Specifying Dimension Data Sources

You can specify whether the shared dimension will be created from a single dimension table, multiple dimension tables, an OLAP data mining model, or from the member properties of another dimension. The dimension table or tables or the mining model you select should contain the column or columns you want in your shared dimension. If you create the dimension from a single dimension table, when you add the shared dimension to a cube and the shared dimension table joins only to the fact table, the cube will have a star schema. If the shared dimension is created from multiple dimension tables, the cube will have a snowflake schema. In either case, you can define a new data source if the table you want to use is not visible in the wizard.

If you create the dimension from an OLAP mining model, you create the dimension from a column that has contents predicted by the mining model, and the mining model dimension has the same impact on a cube's schema as dimensions created from tables when it is added to the cube.

If you use the member properties of another dimension, you can create a virtual dimension. A virtual dimension can be used like other dimensions when it is added to a cube.

Another type of dimension you can create when using a single dimension table is a parent-child dimension. Two columns of the dimension table are used to define the dimension where one column identifies each dimension member, and the other defines its parent. You can also define a new data source for a parent-child if needed.

Steps Required in Some Situations

The following steps are required in certain situations, which are described here.

Select the Dimension Type

If the dimension table contains columns that contains both date-formatted columns and columns of other formats, you specify whether the shared dimension is a standard or time dimension.

Create Time Dimension Levels

If you specified the creation of a time dimension, you also specify the levels for the time dimension.

Create and Edit Joins

If you are creating a snowflake-schema dimension, you confirm, delete, or specify new joins between dimension tables.

Select Levels

For star-schema, snowflake-schema, or virtual dimensions, you define the levels to be used in the dimension. Each level is defined by a column (except in some time dimensions).

Levels within a regular shared dimension are organized hierarchically. For example, in a Location dimension, the levels are Country, State, and City. The values (that is, members) in each level determine the column and row headings that end users see when they browse the cube with a tabular browser. (In graphical browsers, end users see other types of descriptive labels produced by the members. For example, each member produces a value on the scale of an axis.) The column you select is used to supply member names and member keys.

Specify Member Key Columns

For star-schema or snowflake-schema dimensions, you can change the member key columns if the default values supplied by the wizard do not uniquely identify level members.

Select Columns for Parent-Child Dimension

If you are creating a parent-child dimension, you select the two columns upon which the parent-child dimension is based.

Select the Dimension with Member Properties

If you are creating a virtual dimension, you select the source dimension containing the member that you will use to define your dimension.

Select Levels for Virtual Dimension

If you are creating a virtual dimension, you also select the member properties from the source dimension to define the levels of your dimension.

Select Mining Model and Predictable Column

This step appears only if you are using an OLAP mining model to define a data mining dimension. You select the mining model and one of its available predictable columns to create the dimension.

Optional Features

Advanced options in the Dimension Wizard allow you to further tailor your shared dimension. Depending upon the type of dimension being created, these are accessible by selecting them from the **Select Advanced Options** step of Dimension Wizard. Another option, **Create a hierarchy of a dimension**, is available in the last step of the wizard. If you started the Dimension Wizard from Cube Editor or the Cube Wizard, you can change the new dimension, which is a private dimension by default in this case, to a shared dimension.

Advanced options that can be enabled include the following:

Changing dimension

If you are creating dimensions from single or multiple tables, you can make the dimension a changing dimension. Compared to other dimensions, a changing dimension allows more kinds of changes without the necessity of fully processing the dimension or the cubes that contain it.

Custom rollups

For parent-child dimensions, you can define formulas or mathematical operators that determine how members are aggregated.

Members with data

For parent-child dimensions, you can enable associated data for dimension members.

Ordering and uniqueness of members

For all dimension types except data mining dimensions, you can enable member sorting. Select from member name, member key, or the contents of a specified column. For each level, you can also specify the scope of uniqueness among member names and member keys.

Storage mode and members groups

If you are creating dimensions from single or multiple tables, you can determine the storage mode for dimension members and enable member grouping. You can specify whether the dimension has a storage mode of MOLAP or ROLAP. The preferred storage mode is MOLAP, but it is not supported in extremely large dimensions, that is, those generally having 5 million members or more. If you select MOLAP, you can specify that member groups be created in a level inserted automatically above the dimension's lowest level. This occurs only if the lowest level contains more than 64,000 members under a common parent.

Writeback

For parent-child dimensions, you can enable writeback capability. Changes to dimension members are written directly to the dimension table.

As mentioned previously, the **Create a hierarchy of a dimension** option is available in the last step of the wizard. This option is usually selected to create one hierarchy of a multiple-hierarchy dimension. To create another hierarchy within the dimension, run the wizard again and specify the same dimension name but a different hierarchy name.

To create a shared dimension using the Dimension Wizard

Creating a Shared Dimension with the Editor

You can use Dimension Editor to create new shared dimensions such as regular dimensions from single or multiple relational tables or virtual dimensions; you can also create new shared dimensions with multiple hierarchies in Dimension Editor. However, you cannot use Dimension Editor to create new parent-child dimensions or data mining dimensions; you must create these using the Dimension Wizard.

In Dimension Editor, you select several options for the shared dimension: data source, dimension table, and dimension levels.

The data source that you select should contain the tables and columns that define the shared dimension. A data source name identifies a database resource and parameters for its usage.

The dimension table or tables that you select should contain the column or columns you want in your shared dimension. A dimension table is a table that contains a dimension's members. It is a peripheral table joined to a central fact table in a cube's schema.

Select the levels you want in your shared dimension. Each level is defined by a column. Levels within a regular shared dimension are organized hierarchically. For example, in a Location dimension, the levels are Country, State, and City. The values (that is, members) in each level determine the column and row headings that end users see when they browse the cube with a tabular browser. (In graphical browsers, end users see other types of descriptive labels produced by the members. For example, each member produces a value on the scale of an axis.) You can modify the properties of the dimension and its levels.

To create a shared dimension using Dimension Editor

Creating Virtual Dimensions

Virtual dimensions enable you to display categorical labels from a single dimension table on multiple axes in browsers without increasing cube size. (Without virtual dimensions, multiple regular or parent-child dimensions would be required.) Specifically, you can display the members of a dimension level on one axis and an associated attribute (that is, a member property) of those members on the other axis. This presentation is useful when end users want to explore the trends of measures depending on the relationship between members and member properties.

For example, an end user requests to analyze Sales by Product Name and Package Material to explore customers' preferences for recyclable Package Materials. If the Product dimension already contains Product Name, add Package Material as a member property and create a virtual dimension from it. After you add the virtual dimension to the cube, the user can cross-reference Product Names and Package Materials and compare the Sales at the intersections.

The main advantages of virtual dimensions are storage savings and reduced cube processing time. Aggregation data for virtual dimensions is not stored. It is calculated in memory. Therefore, if you add a virtual dimension to a cube, its storage usage and processing time do not increase as when you add a regular or parent-child dimension. All cubes require dimensions, but not all cubes require virtual dimensions.

A virtual dimension is created from member properties or columns of another dimension that is not a virtual dimension. You can add a virtual dimension to a cube only if the dimension that supplies the member properties or columns is also included in the cube. To create a virtual dimension from columns, the columns must be in one of the other dimension's tables, but they do not have to be part of the dimension's definition (for example, the source of a level). For more information about member properties, see Creating Member Properties.

To create a virtual dimension based on member properties, use the Dimension Wizard. The wizard allows you to select the member properties that define the virtual dimension. Virtual dimensions based on member properties cannot be created in Dimension Editor. They can be created in Cube Editor only if you start the Dimension Wizard from within Cube Editor. However, after it is created, a virtual dimension can be edited in Dimension Editor (if the virtual dimension is shared) or Cube Editor (if the virtual dimension is private).

To create a virtual dimension based on member properties

Viewing Dimension Data

Dimension Browser allows you to view dimension data within Analysis Manager. You can view the members, member property values, and custom member formulas of a dimension.

If the dimension is write-enabled, you can also update these items.

To browse a shared dimension

Analysis Services

Building and Processing Cubes

Building and processing cubes are two of the most common administrative tasks in Microsoft® SQL Server™ 2000 Analysis Services. Analysis Manager provides wizards and editors to help you perform these tasks. When you build a cube, you should consider the data source of the cube, the type of storage the cube will use, and how much data you want to precalculate.

When you process a cube, you can choose from different processing methods depending on the type of changes you want to incorporate. This topic contains information to help you make these decisions.

You can also use Decision Support Objects (DSO) to build and process cubes programmatically. For more information, see Using Decision Support Objects.

Topic	Description
Building Cubes	Describes how you can build and maintain cubes using the Cube Wizard and Cube Editor, and the items you specify in each step of the process.
Designing Storage Options and Aggregations	Describes the steps and options you use when you design storage options and aggregations using the Storage Design Wizard.
Managing Linked Cubes	Discusses the special requirements of linked cubes, including synchronization and the use of user-defined functions.
Processing Cubes	Describes conditions for performing complete processing on cubes.
Viewing Cube Data	Describes how to view cube data using Cube Browser, and also reviews changing dimensions, drilling down, and slicing.

See Also

Cubes

Processing Cubes

Building Cubes

You can build a cube using the Cube Wizard or Cube Editor. The Cube Wizard takes you through the process in a series of steps. Cube Editor allows you to perform some of the steps in your own order. To build a cube using either the Cube Wizard or Cube Editor, you must specify the data source, fact table, measures, and dimensions for your cube.

The data source contains the fact table and dimension tables you want to include in your cube. A data source name identifies a database resource and parameters for its usage.

The fact table contains the measures you want to include in your cube. A fact table is the central table in a schema. It contains the numerical data (that is, measures) of main interest to end users of the cube. A fact table also contains foreign keys that are joined to primary keys in dimension tables.

The measures that you select are the ones that you want to make available to end users. A measure contains numerical data (for example, Sales) viewed and analyzed by end users. Each measure corresponds to a column in the fact table. This column supplies the measure's values.

The dimensions that you select will also be made available to end users. Dimensions are descriptive categories by which the measures can be separated for analysis. In tabular browsers, dimensions provide the column headings, row headings, and subheadings by which the measures are separated and displayed to end users. (In graphical browsers, they provide other types of descriptive labels but with the same function as in tabular browsers.) For example, the measure is Sales, and the dimensions are Time, Location, and Product. End users can separate Sales into various categories of Time, Location, and Product. Time provides headings for individual years and subheadings for months. Location and Product also supply a variety of headings and subheadings.

Each dimension is created from one or more columns in a dimension table. These columns supply the values of the dimension, and produce the column headings, row headings, and subheadings seen by end users.

Each dimension table contains a primary key that is joined to a foreign key in either the fact table or another dimension table.

You can also build virtual cubes, which combine elements of multiple, previously built cubes. When end users browse the virtual cube, they see the combined elements together as if they were in a single cube. One of the advantages of virtual cubes is that their definitions, but not their data, are stored. Thus, virtual cubes require much less storage space than regular cubes.

You can also build linked cubes, which can provide additional flexibility in distributing cube data to end users.

See Also

Cubes

Cube Structure

Cube Wizard

Cube Editor - Data View

Cube Editor - Schema View

Data Sources

Dimensions

Measures

Building a Cube with the Wizard

To build a cube quickly and easily, use the Cube Wizard.

To start the Cube Wizard

Building a Cube with the Editor

Cube Editor allows you to choose the order in which you perform certain tasks and variations of those tasks. You can also see and modify object properties. You can also use Cube Editor to update existing cubes.

To build a cube with Cube Editor

Creating and Maintaining Private Dimensions

A private dimension is used only in one cube and cannot be used in other cubes. To create a private dimension, use Cube Editor and the Dimension Wizard.

To create a private dimension

Adding a Multiple-Column Measure to a Cube

A measure can contain multiple columns combined in an expression. For example, the Profit measure is the difference of two numeric columns: **Sales** and **Cost**.

When you build or update a cube in Cube Editor, you can add such a measure.

To add a multiple-column measure to a cube

Building a Virtual Cube

To build a virtual cube quickly and easily, use the Virtual Cube Wizard. The wizard takes you through a series of steps to specify cubes, measures, and dimensions used for your virtual cube.

You select the cubes that contain the data you want in your virtual cube. A cube is a set of data organized and summarized into a multidimensional structure defined by measures and dimensions. The virtual cube creates a combined view of your selected cubes by including the measures and dimensions that you select.

The measures you select are the measures that will be available to end users of your virtual cube. A measure contains numerical data (for example, Sales) viewed and analyzed by end users. Each measure corresponds to a column in the fact table of a selected cube. This column supplies the values of the measure.

The dimensions you select are the dimensions that will be available to end users of your virtual cube. Dimensions are descriptive categories by which the measures can be separated for analysis. In tabular browsers, they provide the column headings, row headings, and subheadings by which the measures are separated and displayed to end users. (In graphical browsers, they provide other types of descriptive labels but with the same function as in tabular browsers.) For example, if the measure is Sales, and the dimensions are Time, Location, and Product, end users can separate Sales into the categories of Time, Location, and Product. Time provides headings for individual years and subheadings for months. Location and Product also supply a variety of headings and subheadings.

Each dimension is created from one or more columns in a dimension table. These columns supply the values of the dimension, which produce the column headings, row headings, and subheadings seen by virtual cube users.

Note A virtual cube based on a linked cube does not support the custom rollup operators, custom rollup formulas, cell calculations, or custom member formulas contained in the linked cube.

To start the Virtual Cube Wizard

Building a Linked Cube

Linked cubes are built using the **Linked Cube** dialog box. This dialog box allows you to specify a source cube for a new linked cube on a publishing server. The source cube can be based on any data source that the current user has permissions on.

To build a new linked cube

Building a Distributed Partitioned Cube

A distributed partitioned cube is a regular cube that utilizes remote partitions, distributing query and processing workload across multiple Analysis servers.

The procedure used to create a distributed partitioned cube starts with the creation of a regular cube. To create a regular cube, use either the Cube Wizard or Cube Editor.

To start the Cube Wizard

Building a Real-Time Cube

A real-time cube is a regular cube that utilizes relational OLAP (ROLAP) shared dimensions, private dimensions, or partitions enabled for real-time updates. Real-time cubes support real-time OLAP in Microsoft® SQL Server™ 2000 Analysis Services.

Note Real-time OLAP features require a SQL Server 2000 data source to be used when creating the cube. To use real-time updates, an Analysis server must use a system administrator account to connect to SQL Server.

If you want to use a ROLAP shared dimension enabled for real-time updates, it is recommended that you create the dimension before creating the real-time cube. For more information about creating shared dimensions, see Creating Shared Dimensions.

The procedure for creating a real-time cube begins with the creation of a regular cube. To create a regular cube, use either the Cube Wizard or Cube Editor.

To start the Cube Wizard

Designing Storage Options and Aggregations

Use the Storage Design Wizard to quickly and easily set storage options and design aggregations for a partition. The wizard operates on a single partition at a time so that you can select different options and designs for each partition. If you start the wizard by selecting a multiple-partition cube, the wizard prompts you to select a partition. You can also start the wizard by selecting a single-partition cube or a partition. The wizard takes you through steps to specify storage and aggregation options for a partition.

Select a storage option if no aggregations exist or if you choose to replace existing aggregations. Each option is briefly described in the following table.

Storage option	Description
MOLAP	Multidimensional OLAP (MOLAP) stores aggregations and a copy of the partition's source data in a multidimensional structure on an Analysis server computer.
ROLAP	Relational OLAP (ROLAP) stores aggregations in a relational structure and leaves the partition's source data in its existing relational structure.
HOLAP	Hybrid OLAP (HOLAP) stores aggregations in a multidimensional structure on an Analysis server computer and leaves the partition's source data in its existing relational structure.

Each storage option has advantages and disadvantages. For more information, see Partition Storage.

Aggregations are precalculated summaries of cube data that help enable Microsoft® SQL Server™ 2000 Analysis Services to provide rapid query responses. Select a method of controlling the number of aggregations the wizard will design, and then let the wizard design the aggregations.

The goal is to design the optimal number of aggregations. This number should not only provide satisfactory response time, but also prevent excessive partition size. A greater number of aggregations produces faster response time but it also requires more storage space. Moreover, as the wizard designs more and more aggregations, earlier aggregations produce considerably larger performance gains than later aggregations. You can control the number of aggregations the wizard designs by one of the following methods available in the wizard:

- Specify a storage space limit for the aggregations.
- Specify a performance gain limit.
- Stop the wizard manually when the displayed **Performance vs. Size** curve starts to level off at an acceptable performance gain.

For more information about aggregations, see Aggregations.

The final step of the wizard allows you to process or defer processing. Processing creates the aggregations you design with the wizard, while deferring processing saves the designed aggregations for future processing, thus allowing design activities to continue without having to process. Depending on the size of the partition, processing may take considerable time.

To start the Storage Design Wizard

Processing Cubes

When you process a cube, the aggregations designed for the cube are calculated and the cube is loaded with the calculated aggregations and data. Processing a cube involves reading the dimension tables to populate the levels with members from the actual data, reading the fact table, calculating specified aggregations, and storing the results in the cube. After a cube has been processed, users can query it.

Caution Referential integrity of the data warehouse is not verified by Microsoft® SQL Server™ 2000 Analysis Services during processing. For example, if the cube's fact table contains foreign key values that are not present in a joined dimension table's primary key column, the rows that contain those values are not processed. In this case, processing does not produce an error message, but the cube contains incomplete and, therefore, inaccurate data.

Before attempting to browse the cube, you must process the cube if you perform any of the following tasks:

- Building the cube and designing its storage options and aggregations.
- Changing the cube's structure (measures, dimensions, and so on) and saving the changes to the cube.
- Changing the structure of a shared dimension used in the cube.

Also, if data in the cube's data warehouse has been added or changed, processing is recommended in order to ensure accurate results when browsing the cube.

Note Newly processed cubes are visible to end users only after they reconnect to the server computer.

Cube Processing Options

Each of the following three processing options is appropriate in different circumstances:

- **Full Process**
- **Incremental update**
- **Refresh data**

In addition to these three mutually exclusive options, you can select a fourth option, **Incrementally update the dimensions of this cube**, in conjunction with any of these options. This option allows you to incrementally update the cube's dimensions as part of the cube processing.

These options are available in the **Process a Cube** dialog box, which is displayed when you right-click a cube in the Analysis Manager tree pane and then click **Process**.

To process a cube

Managing Linked Cubes

After a linked cube has been created and processed, it is managed the same way as a normal cube. The only exception to this rule as that linked cubes cannot be write-enabled. A linked cube contains all of the information of the parent cube. All of the cube's meta data, such as actions and commands, is available to the subscribed linked cube. However, you must ensure that all user-defined functions used by the publishing cube are distributed to subscribing cubes.

Note Because user-defined functions are difficult to maintain across an enterprise-wide application, published cubes should not use user-defined functions.

Linked cubes and their source cubes are synchronized whenever the linked cube is processed. Linked cubes are automatically synchronized whenever their source cube is changed. This update happens the first time the linked cube connects to a source cube that has changed since the last time the link cube connected. In some cases, the change may not be detected immediately. In this situation, you can explicitly process the linked cube to update it.

For more information, see Working with Linked Cubes.

See Also

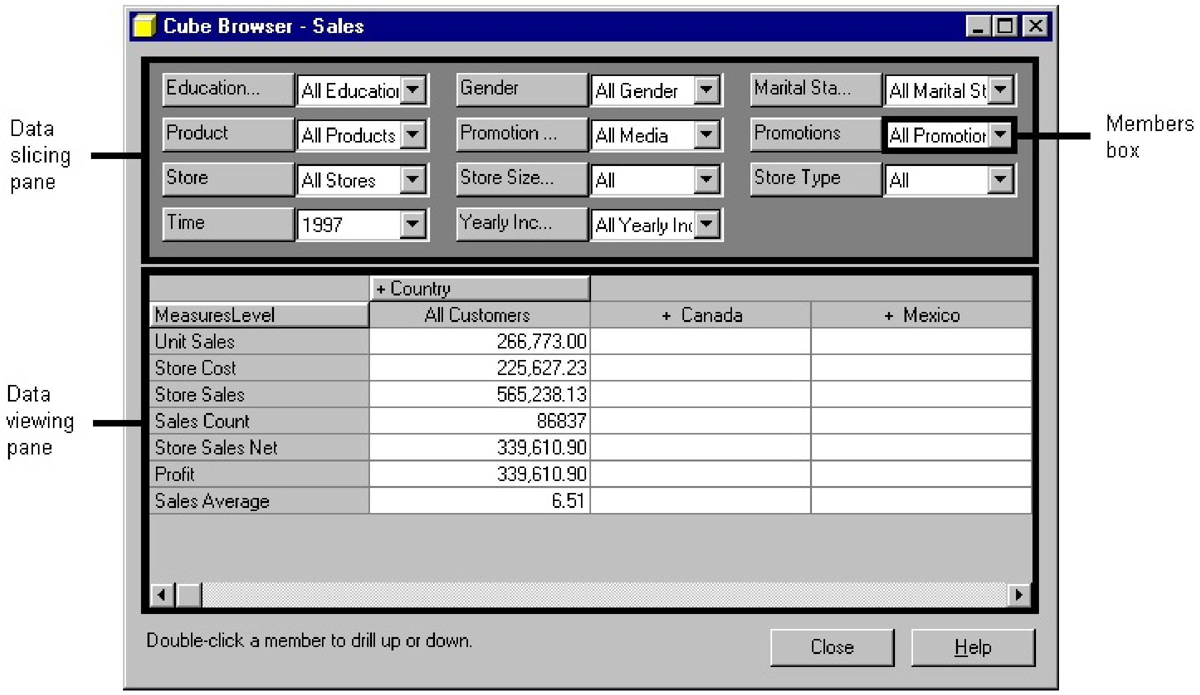
Linked Cubes

Building a Linked Cube

Viewing Cube Data

Analysis Manager provides Cube Browser for rapid and easy access to your cube data. Cube Browser allows you to quickly browse multidimensional data in a flattened, two-dimensional grid format.

Cube Browser appears when you right-click a processed cube and then click **Browse Data**, or when you click **Browse Sample Data** in the last step of the Cube Wizard.



Education...	All Educatio	Gender	All Gender	Marital Sta...	All Marital St
Product	All Products	Promotion ...	All Media	Promotions	All Promotio
Store	All Stores	Store Size...	All	Store Type	All
Time	1997	Yearly Inc...	All Yearly Inc		

	+ Country		
MeasuresLevel	All Customers	+ Canada	+ Mexico
Unit Sales	266,773.00		
Store Cost	225,627.23		
Store Sales	565,238.13		
Sales Count	86837		
Store Sales Net	339,610.90		
Profit	339,610.90		
Sales Average	6.51		

Double-click a member to drill up or down.

Cube Browser functions identically to the **Data** tab in Cube Editor. An exception is that you cannot use Cube Browser to view sample data for an unprocessed cube unless you display Cube Browser from the Cube Wizard. If you right-click an unprocessed cube and then click **Browse Data**, Cube Browser does not present any data for viewing. For more information about browsing unprocessed cubes and the **Data** tab of Cube Editor, see Browsing an Unprocessed Cube and Cube Editor - Data View.

Browsing Cube Data

When you browse cube data, you can view different dimensions, drill down into members, and slice through dimensions.

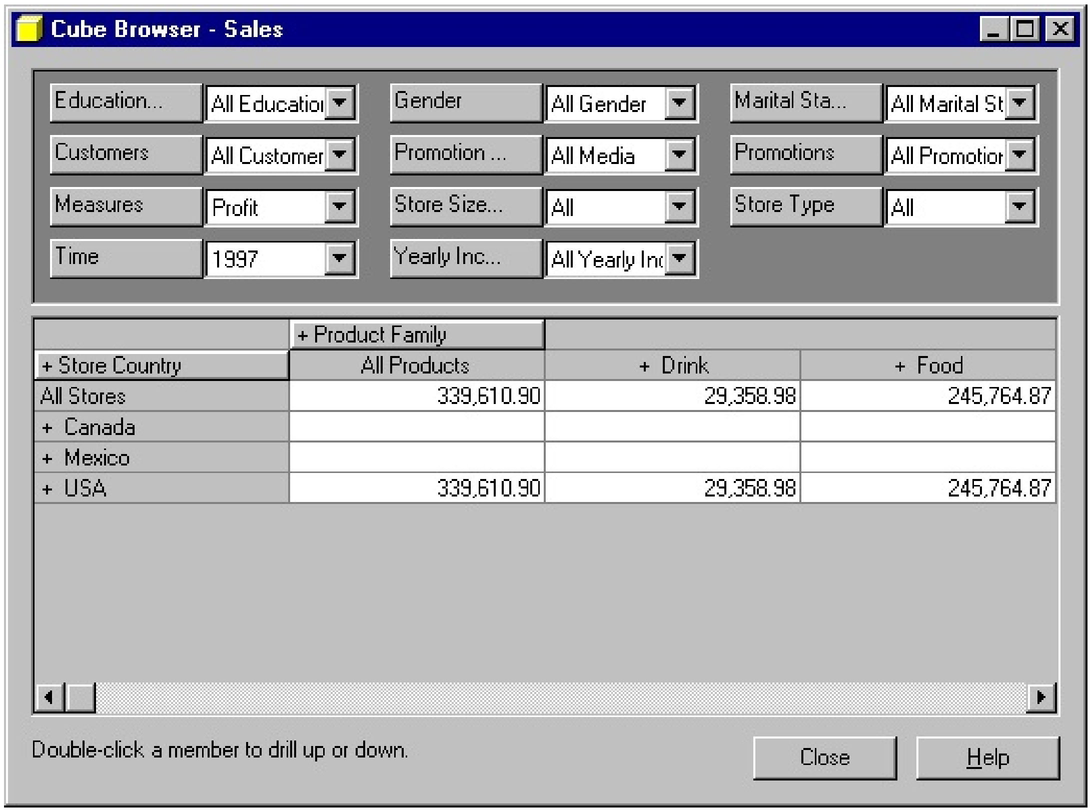
Changing the Dimensions

You can quickly change the data viewing pane by dragging a dimension from the data slicing pane to the data viewing pane. For example, using the Sales cube in the sample **FoodMart 2000** database, to view profit totals by store location for each product category, perform the following steps:

1. Drag the Store dimension to the Measures dimension located on the row axis.

2. Drag the Product dimension to the Customers dimension located on the column axis.

Cube Browser should now look like this.


Cube Browser - Sales

Education...	All Educatio	Gender	All Gender	Marital Sta...	All Marital St
Customers	All Customer	Promotion ...	All Media	Promotions	All Promotior
Measures	Profit	Store Size...	All	Store Type	All
Time	1997	Yearly Inc...	All Yearly Inc		

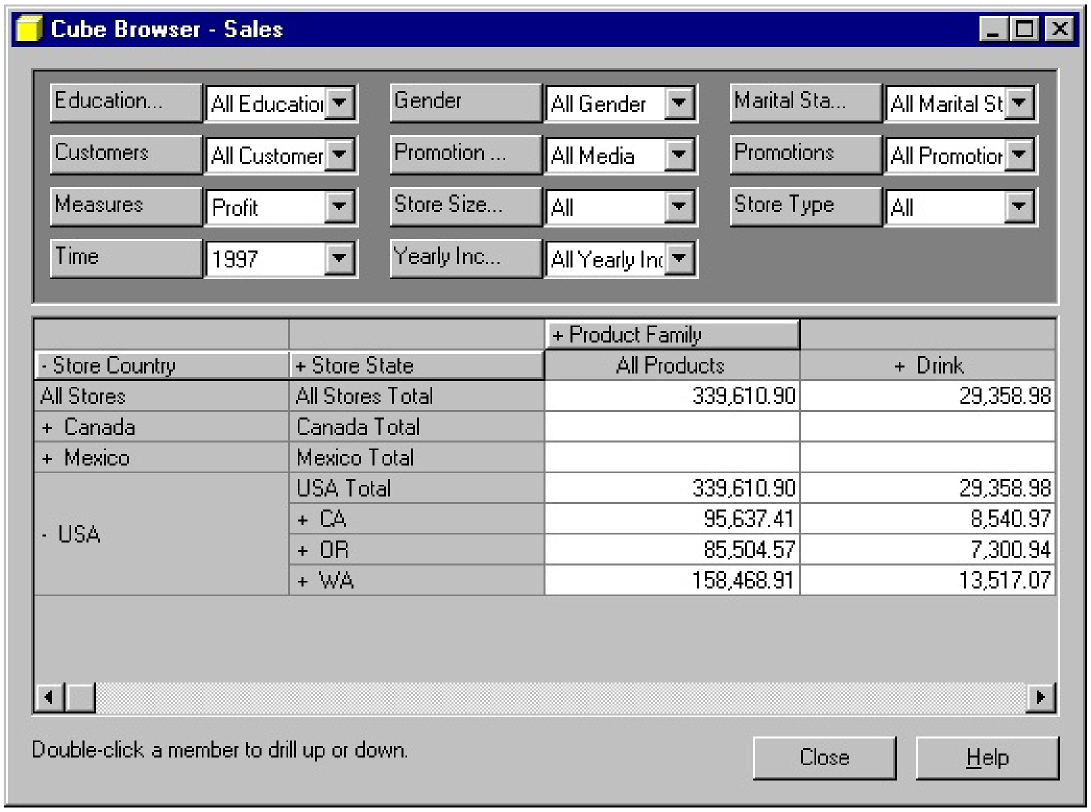
	+ Product Family		
+ Store Country	All Products	+ Drink	+ Food
All Stores	339,610.90	29,358.98	245,764.87
+ Canada			
+ Mexico			
+ USA	339,610.90	29,358.98	245,764.87

Double-click a member to drill up or down.

For more information about these kinds of operations, see Cube Browser.

Drilling Down into Members

To drill down into a particular member, double-click the member. In the previous example, to drill down into the details of the USA store profit numbers, double-click **USA** on the row axis. Cube Browser should now look like this.



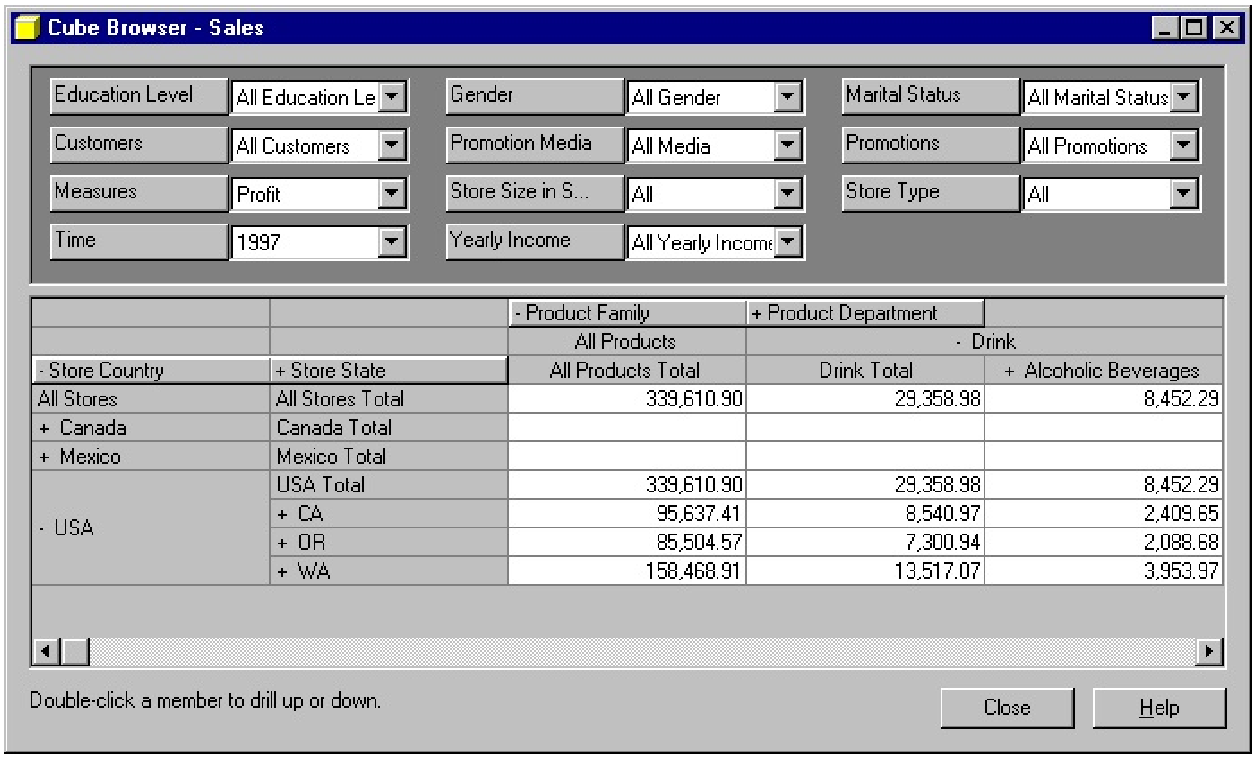
Cube Browser - Sales

Education...	All Educatio	Gender	All Gender	Marital Sta...	All Marital St
Customers	All Customer	Promotion ...	All Media	Promotions	All Promotio
Measures	Profit	Store Size...	All	Store Type	All
Time	1997	Yearly Inc...	All Yearly Inc		

		+ Product Family	
- Store Country	+ Store State	All Products	+ Drink
All Stores	All Stores Total	339,610.90	29,358.98
+ Canada	Canada Total		
+ Mexico	Mexico Total		
- USA	USA Total	339,610.90	29,358.98
	+ CA	95,637.41	8,540.97
	+ OR	85,504.57	7,300.94
	+ WA	158,468.91	13,517.07

Double-click a member to drill up or down.

To further drill down into USA store profit by individual drink categories, double-click **Drink** on the column axis. Cube Browser should now look like this.



Cube Browser - Sales

Education Level	All Education Le	Gender	All Gender	Marital Status	All Marital Status
Customers	All Customers	Promotion Media	All Media	Promotions	All Promotions
Measures	Profit	Store Size in S...	All	Store Type	All
Time	1997	Yearly Income	All Yearly Income		

		- Product Family	+ Product Department	
		All Products	- Drink	
- Store Country	+ Store State	All Products Total	Drink Total	+ Alcoholic Beverages
All Stores	All Stores Total	339,610.90	29,358.98	8,452.29
+ Canada	Canada Total			
+ Mexico	Mexico Total			
- USA	USA Total	339,610.90	29,358.98	8,452.29
	+ CA	95,637.41	8,540.97	2,409.65
	+ OR	85,504.57	7,300.94	2,088.68
	+ WA	158,468.91	13,517.07	3,953.97

Double-click a member to drill up or down.

Drill down as deep into your cube data as the levels or Cube Browser memory allows.

Cube Browser has an internal memory limit, which you may reach if you attempt to browse too much data or drill down too deeply. When you reach the limit, the following message is displayed:

```
Unable to display current view of cube.
Unable to Allocate Memory For Flexgrid.
```

The limit cannot be increased by adding or allocating more memory. If you reach the limit, reduce the amount or depth of data you are attempting to browse or use another browser.

Slicing Through Cube Dimensions

To filter the cube data, select a member from a members box. In the preceding example, to view the data for only small grocery stores, click the **Store Type** members box, expand the members, and then click **Small Grocery**.

Slicing Through Time Dimensions

To view small grocery store profit in the second quarter of 1997, click the **Time** members box, expand the members, and then click **Q2** under **1997**.

Browsing an Unprocessed Cube

Cube Browser does not display data for an unprocessed cube. (An exception is that when you create a new cube in the Cube Wizard and click **Browse Sample Data** in the wizard's last step, Cube Browser displays sample data.) Sample data is generated during cube editing sessions so that you can view the impact of your changes without having to process the cube after each change.

If a cube has not been processed since it was last changed, the following message appears when you open Cube Browser by right-clicking a cube and then clicking **Browse Data**:

```
Unable to browse the cube 'cube-name'.
Cube not processed. To browse sample data for this cube, open
Cube Editor, and then on the View menu, click Data.
```

If you receive this message, you have two options.

First, you can close Cube Browser, process the cube, and then browse it with Cube Browser. However, cube processing can take considerable time. For more information about processing cubes, see Processing Cubes.

Second, you can close Cube Browser and browse sample data in Cube Editor. Browsing sample data enables you to preview the structure of a cube without viewing its actual data.

To browse sample data in Cube Editor

Analysis Services

Creating Security Roles

The following topics describe how to enable end users to access cube data through client applications. For information about enabling administrators to access cube data and meta data and to perform administrative functions, see Administrator Security.

Topic	Description
Creating Database Roles	Identifies ways of creating database roles.
Creating Cube Roles	Describes ways of creating cube roles, changing their default values, and specifying their cell security.
Creating Mining Model Roles	Describes ways of creating mining model roles and changing their default values.
Defining Custom Rules for Dimension Security	Describes creating custom rules in database and cube roles.
Defining Custom Rules for Cell Security	Describes ways of creating custom rules.

Microsoft® SQL Server™ 2000 Analysis Services uses Microsoft Windows NT® 4.0 or Microsoft Windows® 2000 user accounts and groups to define roles for end user access to Analysis Services databases and cube data. Essentially, you combine user accounts and groups into roles and then assign the roles to cubes. Set up these user accounts and groups in Windows NT 4.0 User Manager or Windows 2000 Computer Management before you create roles in Analysis Services.

The three types of roles included in Analysis Services are database, cube, and mining model roles.

The database role is defined at the Analysis Services database level; it can be assigned to multiple cubes in the database, thereby granting the role's users access to these cubes. Such an assignment creates a cube role with the same name as the database role. A database role provides defaults for cube roles of the same name. Use Database Role Manager to maintain database roles.

The cube role is created at the cube level when a database role is assigned to a cube; a cube role applies to only that cube. Defaults in a cube role are derived from the database role of the same name, but some of these defaults can be overridden in the cube role. A cube role contains additional options such as cell security that are not contained in a database role. Use Cube Role Manager to maintain cube roles.

To implement cube-specific security using roles, perform two procedures:

1. Create database roles by combining user accounts and groups and specifying the kind of access the roles are allowed.

2. For each cube in the database, create cube roles by selecting the database roles that can access the cube. You can then change the defaults of the cube roles. These defaults are provided by the database roles.

The mining model role is created at the mining model level when a database role is assigned to a mining model; a mining model role applies to only that mining model. Defaults in a mining model role are derived from the database role of the same name, but some of these defaults can be overridden in the mining model role.

The default access provided by roles is read (that is, read-only), but you can also grant read/write access to select database or cube roles if a cube or dimension is write-enabled.

A write-enabled cube allows users in roles with read/write access to save changes to the cube's data. However, because the changes are saved separately from the original cube data, they affect only displayed cube data and can be deleted if necessary. The separately stored changes are called writeback data. A write-enabled dimension allows users in roles with read/write access to update the dimension's members. These changes are recorded directly in the dimension table.

See Also

Cube Role Manager

Database Role Manager

Database, Cube, and Mining Model Roles

Data Mining Models

Maintaining Write-Enabled Cubes and Writeback Data

Mining Model Role Manager

Security and Authentication

Write-Enabled Cubes

Write-Enabled Dimensions

Creating Database Roles

After you have defined the necessary Microsoft® Windows NT® 4.0 or Windows® 2000 user accounts and groups, you can create database roles.

Note Some values in a database role can be overridden in cube roles of the same name.

To create a database role, use Database Role Manager and the **Database Role** dialog box.

How to create a database role

Creating Cube Roles

For each cube in the database, select the database roles that can access it. Each selection creates a cube role with the same name as the database role. A cube role's defaults are derived from the selected database role, but you can change some of the default values in the cube role.

Alternatively, you can create a cube role without selecting an existing database role. This action creates a database role with the same name as the new cube role.

In a cube role, you can also specify cell security, which cannot be defined in database roles. Specifying cell security is optional.

To create a cube role, change its default values, and specify cell security, use Cube Role Manager and the **Cube Role** dialog box.

To create a cube role, change its default values, and specify cell security

Creating Mining Model Roles

To create mining model roles, for each mining model in the database, select the database roles that can access the model. Each selection creates a mining model role with the same name as the database role. A mining model role's defaults are derived from the selected database role, but you can change some of the default values in the mining model role.

Alternatively, you can create a mining model role without selecting an existing database role. This action creates a database role with the same name as the new mining model role.

To create a mining model role and change its default values, use Mining Model Role Manager and the **Mining Model Role** dialog box.

To create a mining model role and change its default values

Defining Custom Rules for Dimension Security

A custom rule is the most flexible type of rule for controlling access to dimension members by users in a role. In a custom rule, you can allow and deny access to specific dimension levels and members. You can also control visual totals and select a default member.

You can create a custom rule for dimension security in a database role or in a cube role. Custom rules in a cube role override custom rules in the database role of the same name.

To create a custom rule in a database role, use Database Role Manager, the **Database Role** dialog box, and the **Custom Dimension Security** dialog box.

To create a custom rule for dimension security in a database role

Defining Custom Rules for Cell Security

A custom rule is the most flexible type of rule for controlling access to cube cells by users in a cube role. In a custom rule, you can allow or deny access to any combination of cube cells. For each cell permission in the cube role, you can allow access to some cells and deny access to others.

To create a custom rule for cell security, use Cube Role Manager, the **Cube Role** dialog box, and the **Cube Cell Security** dialog box.

To create a custom rule for cell security

Analysis Services

Managing Partitions

Partitions must be created and managed correctly to avoid inconsistent or inaccurate results. This requirement applies to multiple-partition cubes. It also applies when you incrementally update any cube, including a single-partition cube, because an incremental update creates a temporary partition and merges it into an existing partition.

The integrity of a cube's data relies on the data being distributed among the partitions of the cube such that no data is duplicated among the partitions. When data is summarized from the partitions, any data elements that are present in more than one partition will be summarized as if they were different data elements. This can result in incorrect summaries and erroneous data provided to the end user. For example, if a sales transaction for Product X is duplicated in the fact tables for two partitions, summaries of Product X sales can include a double accounting of the duplicated transaction.

Partitions can be merged; you can use this feature in your overall storage and data update strategy. Partitions can be merged only if they have the same storage mode and aggregation design. To create partitions that are candidates for later merging, you can copy the aggregation design of another partition when you create partitions. You can also edit a partition after it has been created to copy the aggregation design of another partition. Merging partitions must also be performed carefully to avoid duplication of data in the resulting partition, which can cause cube data to be inaccurate.

When you are creating or merging partitions, you may need to perform manual operations on underlying dataor create appropriate filters to ensure that the partitions of the cube always contain the correct data. This topic addresses issues and precautions to be aware of when you are creating or merging partitions or performing incremental updates of cubes.

Topic	Description
Creating Partitions	Contains information about how to partition data using filters or different fact tables without duplicating data.
Merging Partitions	Contains information about how to merge partitions that have different fact tables or different data slices

	without duplicating data.

See Also

Incremental Updates and Partitions

Creating Partitions

You can use the Partition Wizard to create additional partitions in a cube. You can specify which portion of a cube's data is allocated to a partition in two ways:

- Assign the partition a fact table that is different from those used by the cube's other partitions

- Filter data in a fact table used by multiple partitions

For more information, see Different Fact Tables for Partitions and Same Fact Table for Multiple Partitions.

Regardless of the method you use to specify the data for a partition, you must ensure that data is not duplicated among a cube's partitions.

Note You can create multiple partitions in a cube only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

To start the Partition Wizard

Different Fact Tables for Partitions

When you create a partition for a cube, you can choose to use the fact table specified for the default partition of the cube, or you can select a different fact table. All fact tables and dimensions for a cube's partitions must have the same structure as the fact table and dimensions of the cube. For example, different fact tables can have the same structure but contain data for different years or different product lines.

When you use different fact tables for the partitions of a cube, ensure that no data is duplicated among the fact tables. For example, if one fact table contains transactions for only 1997 and another fact table contains transactions for only 1998, they contain independent data. Similarly, fact tables for distinct product lines or distinct geographical areas are independent.

It is possible but not recommended to use different fact tables that contain duplicated data. In this case, you must use filters in the partitions to ensure that data used by one partition is not used by any other partition.

Note Merging partitions that have different fact tables requires special consideration and attention. For more information, see Fact Table Considerations When Merging Partitions.

See Also

Partition Structure

Same Fact Table for Multiple Partitions

When the same fact table is used for more than one partition in a cube, it is important that the same rows are not used in more than one partition. It is possible for a row that is used in more than one partition to be included multiple times when the cube is processed or queried; this can cause queries to return incorrect data.

You can use filters in partitions to ensure that data is not duplicated among the partitions. A partition's filter specifies which data in the fact table is used in the partition. It is important that the filters for all partitions in a cube extract mutually exclusive datasets from the fact table. For example, these filters are mutually exclusive within each set:

Set 1:

```
"SaleYear" = 1997
"SaleYear" = 1998
```

Set 2:

```
"Continent" = 'NorthAmerica'
"Continent" = 'Europe'
"Continent" = 'SouthAmerica'
```

Set 3:

```
"Country" = 'USA'
"Country" = 'Mexico'
("Country" <> 'USA' AND "Country" <> 'Mexico')
```

When you create mutually exclusive filters for partitions, ensure that the combined partition data includes all data you want to include in the cube.

See Also

Partition Filters and Incremental Update Filters

Data Slice

A data slice is an optimization feature that helps direct queries to the data of the appropriate partitions. A data slice is not a substitute for or an alternative to a partition's filter. That is, a data slice should not be used to limit the data selected from the partition's fact table and included in the partition. You can specify a data slice when you create a partition using the Partition Wizard.

A data slice of a partition should reflect as closely as possible the data in the partition. For example, if a partition is limited to 1997 data, the partition's data slice should specify the 1997 member of the Time dimension. It is not always possible to specify a data slice that reflects the exact contents of a partition. For example, if a partition contains data for only January and February, but the levels of the Time dimension are Year, Quarter, and Month, there is no way in the Partition Wizard to select both the January and February members. In such cases, select the parent of the members that reflect the partition's contents. In this example, select Quarter 1.

When the partitions of a cube contain data slices, Microsoft® SQL Server™ 2000 Analysis Services can more efficiently determine which partitions must be accessed to resolve queries on the cube.

Although a data slice sometimes modifies the WHERE clause that is used to populate a partition, this does not always occur. For this reason, whenever it is necessary to limit the data selected from a partition's fact table, you must use a filter, not a data slice.

You cannot slice partitions on a ROLAP dimension.

See Also

Merging Partitions That Have Data Slices

Same Fact Table for Multiple Partitions

Merging Partitions

The ability to merge partitions can be a powerful option, but care must be taken to fully understand both the process of merging and ways to ensure that the merge operation produces your intended results.

To ensure that partitions can be merged at a later time, when you create the partitions, you must take into account certain restrictions on merging partitions. Partitions can be merged only if they meet all the criteria listed here:

- They are in the same cube
- They have the same structure (the default situation)
- They are stored in the same mode (MOLAP, HOLAP, or ROLAP)
- They contain identical aggregation designs

Remote partitions can be merged only with other remote partitions that are defined with the same remote Analysis server.

To create a partition that is a candidate for future merging, when you create the partition in the Partition Wizard, you can choose to copy the aggregation design from another of the cube's partitions. This ensures that these partitions have the same aggregation design; when they are merged, the aggregations of the source partition are combined with the aggregations in the target partition.

To merge two partitions

Fact Table Considerations When Merging Partitions

When you merge partitions, the filters of both partitions are combined using OR to create a filter for the resulting partition. This combined filter specifies the set of facts used in the resulting partition. If the source partition and target partition use the same fact table, the combined filter specifies the fact table data that is appropriate to the resulting partition. Because the facts necessary for the resulting partition are present in the fact table, no further action is necessary.

Important Fact tables are not merged automatically when you merge partitions. You must manually merge fact tables when the partitions being merged have different fact tables.

When you merge partitions that use different fact tables, the resulting partition refers only to the original fact table of the target partition. You must manually merge the facts from the fact table of the source partition into the fact table of the target partition. If this manual step is not performed, the partition does not contain complete information.

MOLAP Partitions

When multidimensional (MOLAP) partitions are merged, the facts stored in the multidimensional structures of the partitions are also merged. This results in an internally complete and consistent partition. However, the facts stored in MOLAP partitions are copies of facts in the fact table. When the partition is subsequently processed, the facts in the multidimensional structure are deleted and data is copied from the fact table as specified by the partition's filter. If the source partition uses a different fact table from the target partition, the fact table of the source partition must be manually merged with the fact table of the target partition to ensure that a complete set of data is available when the resulting partition is processed.

Caution A merged MOLAP partition with an incomplete fact table contains an internally merged copy of fact table data and operates correctly until it is processed.

HOLAP and MOLAP Partitions

When hybrid OLAP (HOLAP) or relational OLAP (ROLAP) partitions that have different fact tables are merged, the fact tables are not automatically merged. Unless the fact tables are manually merged, only the fact table associated with the target partition is available to the resulting partition. Facts associated with the source partition are not available for drilldown in the resulting partition, and when the partition is processed, aggregations do not summarize data from the unavailable table.

Caution A merged HOLAP or ROLAP partition with an incomplete fact table contains accurate aggregations, but incomplete facts. Queries that refer to missing facts return incorrect data. When the partition is processed, aggregations are computed only from available facts.

The absence of unavailable facts might not be noticed unless a user attempts to drill down to a fact in the unavailable table or executes a query that requires a fact from the unavailable table. Because aggregations are combined during the merge process, queries whose results are based only on aggregations return accurate data, whereas other queries may return inaccurate data. Even after the resulting partition is processed, the missing data from the unavailable fact table may not be noticed, especially if it represents only a small portion of the combined data.

Fact tables can be merged before or after merging the partitions. However, the aggregations will not accurately represent the underlying facts until both operations have been completed. It is recommended that you merge HOLAP or ROLAP partitions that access different fact tables when users are not connected to the cube containing these partitions.

Merging Partitions That Have Data Slices

When you merge partitions that have data slices specified in the Partition Wizard, the resulting partition can contain unexpected, incorrect data after it is processed. To prevent this, you can create a filter that specifies the data in the resulting partition.

For example, a cube containing information about three soft drink products has three partitions that use the same fact table. These partitions have data slices based on product. Partition 1 contains data about [ColaFull], Partition 2 contains data about [ColaDecaf], and Partition 3 contains data about [ColaDiet]. If Partition 3 is merged into Partition 2, the data in the resulting partition (Partition 2) is correct and the cube data is accurate. However, when Partition 2 is processed, if it does not have a filter, its content may be determined by the parent of the members at the product level. This parent, [SoftDrinks], also includes [ColaFull], the product in Partition 1. Processing Partition 2 loads the partition with data for all soft drinks, including [ColaFull]. The cube then contains duplicate data for [ColaFull] and returns incorrect data to end users.

In this example, after merging Partition 3 into Partition 2, you can provide a filter such as ("Product" = 'ColaDecaf' OR "Product" = 'ColaDiet') in the resulting Partition 2 to specify that only data about [ColaDecaf] and [ColaDiet] be extracted from the fact table, and that data pertaining to [ColaFull] be excluded. Alternatively, you can specify filters for Partition 2 and Partition 3 when they are created, and these filters will be combined during the merger process. In either case, after the partition is processed, the cube does not contain duplicate data.

See Also

Data Slice

Analysis Services

Enhancing Dimensions with Optional Features

Dimensions can be enhanced with a number of features designed to increase functionality, such as member properties, custom rollup operators, custom member formulas, and member groups. Member properties are a way to associate additional attributes with dimension members. Custom rollup operators allow you to control the way in which aggregate values are calculated from child members. Custom member formulas allow you to specify Multidimensional Expressions (MDX) formulas to specify cell values that override the aggregate values of measures. Member groups provide a way of grouping large numbers of members without adding levels.

Topic	Description
Creating Member Properties	Contains information about member properties and how to use them.
Using Custom Rollup Operators	Describes custom rollup operators and how to use them.
Creating Custom Member Formulas	Describes custom member formulas and the steps to take before you create them.
Creating Member Groups	Contains information about member groups and how to process their dimensions or cubes depending on different property settings.
Creating Dimensions with Multiple Hierarchies	Describes dimensions with multiple hierarchies and the ways to create them.
Adding Hierarchies to Existing Dimensions	Describes the process of adding hierarchies to existing dimensions.

See Also

Custom Rollup Formulas and Custom Member Formulas

Member Groups

Member Properties

Creating Member Properties

Member properties are attributes associated with members. A member is an item in a dimension level. A member property is created in a dimension for the purpose of providing end users with additional information about members. The information is typically less important to end users than the members themselves and does not qualify as a level within the dimension. For example, if a Time dimension contains Year, Month, and Day levels, and the data of the dimension table spans one full calendar year, the Month level contains 12 members: January through December. If the data of the Time dimension table spans two full calendar years, the Month level contains 24 members. Thus, members need not be unique values within a dimension level. Member properties are optional components of dimensions and cubes.

A member property is an attribute associated with a member. For example, each member of the Month level has an associated Boolean member property called Bonus Month. It records whether bonuses are awarded during the month because they are not awarded at a regular frequency. In months in which bonuses are awarded, Bonus Month equals –1, or TRUE; in the remaining months, Bonus Month equals 0, or FALSE.

A member property is stored in a column in the same table as the dimension level containing the members. For example, Bonus Month is stored in the same table as the Month level.

You can create a member property by inserting it into the level that contains the members with which the member property is associated. For example, Bonus Month is defined inside the Month level.

For example, before Bonus Month existed as a member property, it existed as a column in the Time dimension table. End users requested that Bonus Month be added to some cubes. However, it was inappropriate to add Bonus Month as a level because it is not a natural component of the Time dimension hierarchy. If Bonus Month had been added as a level, end users would have drilled down, for example, from Month to Bonus Month to Day, experiencing an awkward and confusing presentation. Because Bonus Month is associated with members of the Month level, the solution was to create the Bonus Month member property by inserting it in the Month level. Consequently, end users can now display Bonus Month values, for example, by right-clicking Month values, and the basic Time dimension hierarchy is undisturbed.

Before you create a member property in a dimension level with a source column containing nonunique values, you must ensure that each member in the level can have only one value for each member property. Otherwise, member property values will be incomplete and misleading because only one value can be displayed for each member. For example, values in the source column of the Salesperson level are not unique. Each Salesperson can have only one Manager, so Manager is a valid member property in the Salesperson level. However, a Salesperson can work in multiple Regions, so Region is not a valid member property. If Region were defined as a member property, it would display only one Region to end users, even for Salespersons who work in multiple Regions.

You can create member properties in both private dimensions and shared dimensions. To create a member property in a private dimension, you can create them in Cube Editor for the cube containing the private dimension, or you can use the Decision Support Objects (DSO) library in a client application that creates an object of **ClassType clsMemberProperty**. To create a member property in a shared dimension, use Dimension Editor.

To create a member property in a shared dimension

Using Custom Rollup Operators

Custom rollup operators provide a simple way to control how member values are rolled up to their parent's values. When custom rollup operators are assigned to the name of a column, either when creating them as an optional feature of new parent-child dimensions in Dimension Wizard or adding them to existing dimensions in Dimension Editor or Cube Editor, the contents of that column are used as the custom rollup operator for the member. This custom rollup operator is applied to the member when evaluating the value of the member's parents.

In Dimension Editor, values for the custom rollup operators are stored in the **Unary Operator Column** property of the level and are applied to each member.

Custom rollup operators provide similar but simplified functionality of custom rollup formulas. In comparison to custom rollup formulas, which use Multidimensional Expressions (MDX) expressions to determine how the members are rolled up, the custom rollup operator uses simple math operators to determine how the value of a member affects the parent. However, the value of the custom rollup operator is unique for each level member, while a custom rollup formula applies to all level members.

In relation to a dimension's custom member formulas, custom member formulas of the preceding level override a level's custom rollup operators. However, the custom rollup operators of a level override the custom rollup expression of the previous level.

The following table lists available custom rollup operators and describes how they behave when they are applied to a level.

Custom rollup operator	Description
+	The value of the member is added to the aggregate value of the preceding sibling members.
-	The value of the member is subtracted from the aggregate value of the preceding sibling members.
*	The value of the member is multiplied by the aggregate

	value of the preceding sibling members.
/	The value of the member is divided by the aggregate value of the preceding sibling members.
~	The value of the member is ignored.

Note Blank values and any other values not found in the table are treated as the plus sign (+) unary operator. There is no operator precedence, so the order of members as stored in the unary operator column determines the order of evaluation. To change the order of evaluation, create a new member property, set its **Type** property to Sequence, and assign sequence numbers corresponding to the order of evaluation in its **Source Column** property.

Besides being able to create new custom rollups for parent-child dimensions in the Dimension Wizard, you can create custom rollups for existing shared or private dimensions.

To create a custom rollup operator for a shared dimension

Creating Custom Member Formulas

Custom member formulas determine the cell values associated with members and override the aggregate functions of measures. They are written in Multidimensional Expressions (MDX). Each custom member formula applies to a single member. Custom member formulas are stored in the dimension table.

Custom member formulas can include MDX functions in Microsoft® SQL Server™ 2000 Analysis Services and some Microsoft Visual Basic® for Applications functions. Custom member formulas can also include some functions from the Microsoft Excel worksheet function library if it is installed. If installed, this library is registered automatically. Other functions are also supported if their libraries are installed and registered. For more information about installing and registering custom function libraries, see Register Function Libraries Dialog Box.

Before you can create custom member formulas for a level, you must first create or select a column in the dimension table to store the custom member formulas. To create or select a column for an existing dimension, use the **Define Custom Member Column** dialog box.

To create or select a column to store custom member formulas

Creating Member Groups

A dimension level can contain member groups, which are system-generated parents of collections of consecutive dimension members. End users see no difference between member groups and ordinary members.

Member groups can provide an intermediate level for drilldown between a level with few members and one with numerous members. They can also be used to satisfy the requirement of having no more than 64,000 members under a parent member.

You can choose to create member groups automatically in regular dimensions while you are creating them with the Dimension Wizard. This option is available only for member groups immediately above the lowest level of the dimension. To select this option, in the **Specify storage mode and create member groups** step of the wizard, select **Create member groups automatically**.

To create member groups in an existing dimension, create a copy of the level that contains the numerous or excess members immediately above the original. Then create member groups in the new level, which you can hide if you don't want end users to see the new level.

IMPORTANT Members of the level used to create member groups must have unique names. Set the **Member Names Unique** property to **True** for the level in Dimension Editor or Cube Editor before you create member groups.

Member groups rely on the **Order By** property of the level that contains their children. This property specifies the sort order of displayed members. Within this sort order, a member group's children are consecutive.

To create member groups in an existing dimension, use Dimension Editor or Cube Editor.

To create member groups

Creating Dimensions with Multiple Hierarchies

Multiple hierarchies can be created for a dimension to provide alternative views of dimension members. For example, a time dimension that has two hierarchies can consist of a regular calendar view and a fiscal calendar view.

In Microsoft® SQL Server™ 2000 Analysis Services, a dimension with multiple hierarchies is actually two or more separate dimensions that can share dimension tables and may share the same aggregations. Using multiple hierarchies helps optimize building aggregations in Analysis Services.

Unlike dimensions with a single hierarchy, the naming schema requires a period to indicate the presence of more than one hierarchy. (Dimensions with a single hierarchy can use a period and a hierarchy name part, but this is optional.) In the previous example, when a time dimension is defined to provide a regular calendar view and a fiscal calendar view, the resulting names can be Time.Calendar and Time.Fiscal.

IMPORTANT When creating dimensions with multiple hierarchies, the hierarchy part of the name should not equal any current or future level name or member name in the dimension because queries using the dimension may be ambiguous.

To help minimize disruption to cubes, it is helpful to identify dimensions with multiple hierarchies before they are deployed. One way to do this is to name the dimension with a period and a hierarchy name part at the time of creation. Additional hierarchies can then be created by using the same dimension name part followed by the period and the hierarchy name part.

Dimensions that have multiple hierarchies can be created in the Dimension Wizard or Dimension Editor. For each hierarchy that is being created, the process is similar to creating a new dimension.

To create a dimension with multiple hierarchies using the Dimension Wizard

Adding Hierarchies to Existing Dimensions

Two approaches can be used to add a hierarchy to an existing dimension. The first can be used if a hierarchy has already been defined for an existing dimension, and the second is used when no hierarchies have been defined. Both approaches involve using Dimension Editor.

The first approach involves editing an existing dimension with a defined hierarchy and saving it as a new dimension with a new hierarchy. The name part of the existing dimension is used followed by a period and a new hierarchy name part.

The second approach uses an existing dimension as the template for new dimensions with multiple hierarchies. If you are adding multiple hierarchies to an existing dimension that has no defined hierarchies, you can base the additional hierarchies on the existing dimension by renaming the dimension with a new dimension name part, a period, and a hierarchy name part. To create the next hierarchy, you use the new dimension name part followed by the period and a different hierarchy name part.

IMPORTANT When creating dimensions with multiple hierarchies, the hierarchy part of the name should not equal any current or future level name or member name in the dimension because queries using the dimension may be ambiguous.

To add a hierarchy to an existing dimension

Analysis Services

Enhancing Cubes with Optional Features

Microsoft® SQL Server™ 2000 Analysis Services offers many powerful optional features you can use to enhance the analysis performed in cubes and the presentation of cube data. There are additional optional features for dimensions that further enhance cube capabilities. For more information, see Enhancing Dimensions with Optional Features.

Topic	Description
Creating Calculated Members	Contains information about using function libraries to create members that display values calculated at run time.
Creating Calculated Cells	Describes using calculated cells to create a multidimensional section of cells, defined by a Multidimensional Expressions (MDX) set expression, to which an MDX value expression is selectively applied depending upon a condition described by an MDX logical expression.
Creating Named Sets	Contains information about different options for creating sets of dimension members or set expressions that you can use in MDX queries.
Creating Actions	Describes different types of actions, which you can create to provide end users with ways to view and even interact with data sources.
Maintaining Write-Enabled Cubes and Writeback Data	Contains information about configuring cube data so that end users can make changes to it.
Specifying Drillthrough Options	Describes ways to allow users to see the source data for a cube cell.

Creating Calculated Members

You can create customized measures or dimension members, called calculated members, by combining cube data, arithmetic operators, numbers, and/or functions. For example, you can create a calculated member called Marks that converts dollars to marks by multiplying an existing dollar measure by a conversion rate. Marks can then be displayed to end users in a separate row or column.

Calculated member definitions are stored, but their values exist only in memory. In the preceding example, values in marks are displayed to end users but are not stored as cube data.

You can create calculated members in regular or virtual cubes. For more information, see Creating Calculated Members in Regular Cubes and Creating and Maintaining Calculated Members in Virtual Cubes.

To create a calculated member, use Calculated Member Builder. It enables you to specify the following options for the calculated member:

- Parent dimension

  Select the parent dimension to include the calculated member. Dimensions are descriptive categories by which the numeric data (that is, measures) in a cube can be separated for analysis. In tabular browsers, dimensions provide the column and row headings displayed to end users when they browse a cube's data. (In graphical browsers, they provide other types of descriptive labels but with the same function as in tabular browsers.) A calculated member provides a new heading (or label) in the parent dimension you select.

  Alternatively, you can include the calculated member in the measures instead of a dimension. This option also provides a new column or row heading, but it is attached to measures in the browser.

- Parent member

  Select a parent member to include the calculated member. This option is unavailable if you select a one-level dimension or **Measures** as the parent dimension.

Dimensions are divided into levels that contain members. Each member produces a heading. While browsing the cube's data, end users can drill down from a selected heading to previously undisplayed subordinate headings. The heading for the calculated member is added at the level directly below the parent member you select.

- Name

  Select the name of the calculated member. This name appears as the column or row heading for the calculated member values when end users browse the cube.

- Value expression

  Specify the expression that produces the values of the calculated member. This expression can be written in Multidimensional Expressions (MDX). The expression may contain any of the following:

  - Data expressions that represent the cube's components such as dimensions, levels, measures, and so on

  - Arithmetic operators

  - Numbers

  - Functions

  IMPORTANT Any calculated member that will be used in the value expression of another calculated member must be created before the calculated member that will use it.

After you create a calculated member, you can rename it. Renaming changes the associated column or row heading displayed to end users.

To rename a calculated member

Creating Calculated Members in Regular Cubes

To create a calculated member in a regular cube, use Cube Editor and Calculated Member Builder.

To create a calculated member using Calculated Member Builder

Creating Calculated Cells

With calculated cells, you can define a Multidimensional Expressions (MDX) formula that can be used to supply a value for each cell in a specific group of cells. Optionally, each cell in the group of cells can be evaluated against a logical MDX statement to determine whether the formula is applied for a particular cell in the group of cells.

For example, you can create a calculated cells definition that provides forecasting values for the next year from a calculation based on the actual values of the current year, but only for specific clients and products. Unlike calculated members, custom members, or custom rollups, this functionality can affect specific cells of a cube, instead of an entire member; a calculated cells definition can be created for only a single cell in an entire cube.

As with calculated members, the definitions for calculated cells are stored. The values for calculated cells are evaluated only when the cube is queried by a client application. Unlike calculated members, however, the cells retain their original underlying data. Thus, calculated cells can be added to a cube without affecting the underlying data, and calculated cells can be removed from the cube without invalidating the cube. Calculated cells are not evaluated when processing a regular or virtual cube; the cube is processed as if the calculated cells definitions do not exist.

With the introduction of calculation passes, calculated cells can be recursively applied across any number of calculation passes, with evaluation starting at a specified calculation pass. For more information about calculation passes, see Understanding Pass Order and Solve Order.

You can create calculated cells in regular or virtual cubes. For more information about creating calculated cells, see Creating Calculated Cells in Regular Cubes and Creating Calculated Cells in Virtual Cubes.

To create a calculated cells definition, use the Calculated Cells Wizard. It enables you to specify the following options for the calculated cells:

- Calculation Subcube

Select the members from each dimension in the cube to define the calculation subcube. The calculation subcube contains the group of cells against which the calculation formula is applied. The combination of all dimensions with specified members and all other dimensions define the calculation subcube. If a given dimension does not have any specified members, all members in that dimension are considered part of the calculation subcube.

If members are not specified from any of the dimensions of the regular or virtual cube, the calculation subcube is defined as the entire regular or virtual cube.

- Calculation Condition

  Define an MDX logical expression (an expression that evaluates to either **True** or **False**) that will be applied to each cell in the calculation subcube. If the calculation condition evaluates to **True** for a cell, the result of the calculation formula applied to that cell is returned when the cell is queried. If the calculation condition evaluates to **False** for a cell, the actual value of the cell is returned when the cell is queried.

  If no calculation condition is specified, the calculation formula will apply to all cells in the calculation subcube.

- Calculation Formula

  Define an MDX expression that will supply the value of each cell in the calculation subcube, subject to the calculation condition. The calculation formula can be any MDX expression that returns a string or numeric value.

After they have been created, calculated cells can also be edited in Cube Editor (for regular cubes) or Virtual Cube Editor (for virtual cubes).

To edit a calculated cells definition in Cube Editor

Creating Calculated Cells in Regular Cubes

To create calculated cells in regular cubes, use Cube Editor and the Calculated Cells Wizard.

To create calculated cells in a regular cube

Creating Calculated Cells in Virtual Cubes

To create calculated cells in virtual cubes, use Virtual Cube Editor and Calculated Cells Wizard.

To create calculated cells in a virtual cube

Creating and Maintaining Calculated Members in Virtual Cubes

You can import calculated members into virtual cubes from their component cubes and you can also create new calculated members directly in virtual cubes. With Calculated Member Builder you can create and edit calculated members in virtual cubes. In a virtual cube, you can import calculated members from the component cubes of that virtual cube so that you do not have to re-create the calculated members. After import, you can edit the calculated member in the virtual cube.

Use Virtual Cube Editor to maintain the calculated members in a virtual cube. Virtual Cube Editor provides single-click access to Calculated Member Builder and the **Import Calculated Members** dialog box, which is used to import calculated members into virtual cubes.

Note Virtual Cube Editor is used for maintaining calculated members in virtual cubes only. To maintain calculated members in regular cubes, use Cube Editor to access Calculated Member Builder.

The topics in the following table describe the three basic procedures for maintaining calculated members in virtual cubes.

Topic	Description
Importing a Calculated Member into a Virtual Cube	Contains information about importing one or more calculated members from the component cubes of a virtual cube to the virtual cube.
Editing a Calculated Member in a Virtual Cube	Describes how to update the calculated members of a virtual cube.
Creating a Calculated Member in a Virtual Cube	Describes how to add new calculated members to a virtual cube.

See Also

Calculated Members

Creating Calculated Members

Virtual Cubes

Virtual Cube Editor

Importing a Calculated Member into a Virtual Cube

When you import a calculated member, you will see an error message if the syntax of the calculated member is incorrect or its value expression includes a measure, dimension, or other object that is not in the virtual cube. (The value expression is displayed beside **Value** in the properties pane of Virtual Cube Editor.) The calculated member is not included in the virtual cube data displayed to end users as long as the error exists. To correct the error, you can:

- Add the missing measure, dimension, or object to the virtual cube and process it. To add a measure or dimension, use Virtual Cube Wizard. To start Virtual Cube Wizard from within Virtual Cube Editor, on the **Edit** menu, click **Structure (Wizard)**.

  -or-

- Edit the calculated member with the error and remove from its value expression the measure, dimension, or object that is not in the virtual cube. For more information, see Editing a Calculated Member in a Virtual Cube.

If you import a calculated member with the same name as a calculated member already in the virtual cube, the imported calculated member is given a new name consisting of its old name and a numeric suffix.

You can import calculated members only from cubes that are in the same database as the edited virtual cube.

To import a calculated member into a virtual cube, use Virtual Cube Editor and the **Import Calculated Members** dialog box.

To import a calculated member into a virtual cube

Editing a Calculated Member in a Virtual Cube

To edit a calculated member in a virtual cube, use Virtual Cube Editor and Calculated Member Builder.

Note Calculated members in virtual cubes are not visible in the Analysis Manager tree pane. To create and access them, you must first display Virtual Cube Editor.

To edit a calculated member in a virtual cube

Creating a Calculated Member in a Virtual Cube

To create a calculated member in a virtual cube, use Virtual Cube Editor and Calculated Member Builder.

Note Calculated members in virtual cubes are not visible in the Analysis Manager tree pane. To create and access them, you must first display Virtual Cube Editor.

To create a calculated member in a virtual cube

Creating Named Sets

A named set is a set of dimension members or a set expression that is created for reuse, for example, in Multidimensional Expressions (MDX) queries. You can create named sets by combining cube data, arithmetic operators, numbers, and/or functions. For example, you can create a named set called Top Ten Factories that contains the ten members of the Factories dimension that have the highest values for the Production measure. Top Ten Factories can then be used in queries by end users. For example, an end user can place Top Ten Factories on one axis and the Measures dimension, including Production, on another axis.

For more information, see Named Sets.

Microsoft® SQL Server™ 2000 Analysis Services provides its own extensive function library for creating named sets. For more information, see MDX Function List. Analysis Services also supports other function libraries, as when you create calculated members. For more information, see Creating Calculated Members.

To create a named set, use Named Set Builder. It enables you to specify the following options for the named set:

Name

Select the name of the named set. This name appears to end users when they browse the cube.

Set expression

Specify the expression that produces the named set. This expression can be written in Multidimensional Expressions (MDX). The expression may contain any of the following:

- Data expressions that represent the cube's components such as dimensions, levels, measures, and so on
- Arithmetic operators

- Numbers

- Functions

  Note If you create the set expression by explicitly naming the members in the set, enclose the list of members in a pair of braces ({}).

For more information, see Named Set Builder.

You can create named sets in regular or virtual cubes. In a virtual cube, you can import named sets from the virtual cube's component cubes so that you do not have to re-create the named sets. After import, you can edit the named set in the virtual cube. For more information, see Creating Named Sets in Regular Cubes and Creating and Maintaining Named Sets in Virtual Cubes.

Creating Named Sets in Regular Cubes

To create a named set in a regular cube, use Cube Editor and Named Set Builder.

Note Named sets in regular cubes are not visible in the Analysis Manager tree pane. To create and access them, you must first display Cube Editor.

To create a named set in a regular cube

Creating and Maintaining Named Sets in Virtual Cubes

You can both import named sets into virtual cubes from their component cubes and create new named sets directly in virtual cubes. You can use Named Set Builder to create and edit named sets in virtual cubes.

Use Virtual Cube Editor to maintain the named sets in a virtual cube. Virtual Cube Editor provides single-click access to Named Set Builder and the **Import Named Sets** dialog box, which is used to import named sets into virtual cubes.

Note Virtual Cube Editor is used for maintaining named sets in virtual cubes only. To maintain named sets in regular cubes, use Cube Editor, from which you can access Named Set Builder.

For more information, see Virtual Cube Editor.

The following topics describe the three basic procedures used to maintain named sets in virtual cubes.

Topic	Description
Importing a Named Set into a Virtual Cube	Describes the procedure you use to copy one or more named sets from a virtual cube's component cubes to the virtual cube.
Editing a Named Set in a Virtual Cube	Describes the procedure you use to update a virtual cube's named sets.
Creating a Named Set in a Virtual Cube	Describes the procedure you use to add new named sets to a virtual cube.

Importing a Named Set into a Virtual Cube

To import a named set into a virtual cube, use Virtual Cube Editor and the **Import Named Sets** dialog box.

To import a named set into a virtual cube

Editing a Named Set in a Virtual Cube

To edit a named set in a virtual cube, use Virtual Cube Editor and Named Set Builder.

Note Named sets in virtual cubes are not visible in the Analysis Manager tree pane. To create and access them, you must first display Virtual Cube Editor.

To edit a named set in a virtual cube

Creating a Named Set in a Virtual Cube

To create a named set in a virtual cube, use Virtual Cube Editor and Named Set Builder.

Note Named sets in virtual cubes are not visible in the Analysis Manager tree pane. To create and access them, you must first display Virtual Cube Editor.

To create a named set in a virtual cube

Creating Actions

An action is an end-user initiated operation upon a selected cube or portion of a cube. The operation can start an application with the selected item as a parameter, or it can retrieve information about the selected item. For more information about actions, see Actions.

To create an action, use the Action Wizard. The wizard takes you through steps to specify the following options for an action:

- Target

  Select the object to which the action is attached. Generally, in client applications, the action is displayed when end users select the target object; however, the client application determines which end-user operation displays actions. Select from the following objects:

  - Cube
  - Dimension
  - Members in a dimension
  - Level
  - Members in a level
  - Cells
  - Named sets
- Type

  Select the type of action. The following table summarizes the available types.

Type	Description
Command Line	Executes a command at the command prompt in Microsoft® Windows NT® 4.0 or Windows® 2000
Statement	Executes an OLE DB command
HTML	Executes an HTML script in an Internet browser
URL	Displays a variable page in an Internet browser
Data set	Retrieves a dataset
Rowset	Retrieves a rowset
Proprietary	Performs an operation using an interface different from those listed earlier in this table

- Syntax

  Specify the parameters that are passed when the action is executed. The syntax must evaluate to a string, and you must include an expression written in Multidimensional Expressions (MDX). For example, your MDX expression can indicate a part of the cube that is included in the syntax. MDX expressions are evaluated before the parameters are passed. Also, MDX Builder is available to help you build MDX expressions.

You can create actions in regular or virtual cubes. In a virtual cube, you can import actions from the virtual cube's component cubes so that you do not have to re-create the actions. After importing these actions, you can edit them in the virtual cube.

See Also

Action Wizard

Creating Actions in Regular Cubes

Creating and Maintaining Actions in Virtual Cubes

MDX Builder

Creating Actions in Regular Cubes

To create an action in a regular cube, use Cube Editor and the Action Wizard.

To create an action in a regular cube

Creating and Maintaining Actions in Virtual Cubes

You can both import actions into virtual cubes from their component cubes and create new actions directly in virtual cubes. You can use the Action Wizard to create and edit actions in virtual cubes.

Use Virtual Cube Editor to maintain the actions in a virtual cube. Virtual Cube Editor provides single-click access to the Action Wizard and the **Import Actions** dialog box, which is used to import actions into virtual cubes.

Note Virtual Cube Editor is used for maintaining actions in virtual cubes only. To maintain actions in regular cubes, use Cube Editor. From Cube Editor you can access the Action Wizard.

For more information, see Virtual Cube Editor.

There are three basic procedures for maintaining actions in virtual cubes. These procedures are described in the following topics.

Topic	Description
Importing an Action into a Virtual Cube	Describes the procedure for copying one or more actions from the component cubes of a virtual cube to the virtual cube itself
Editing an Action in a Virtual Cube	Describes the procedure for updating actions for a virtual cube
Creating an Action in a Virtual Cube	Describes the procedure for adding new actions to a virtual cube

Importing an Action into a Virtual Cube

To import an action into a virtual cube, use Virtual Cube Editor and the **Import Actions** dialog box.

To import an action into a virtual cube

Editing an Action in a Virtual Cube

To edit an action in a virtual cube, use Virtual Cube Editor and the Action Wizard.

How to edit an action in a virtual cube

Creating an Action in a Virtual Cube

To create an action in a virtual cube, use Virtual Cube Editor and the Action Wizard.

How to create an action in a virtual cube

Maintaining Write-Enabled Cubes and Writeback Data

If you write-enable a cube, end users can change displayed cube data while they browse it. However, the changes are saved in a separate table called a writeback table, not in the cube data or source data. End users who browse a write-enabled cube see the net effect of all changes in the writeback table for the cube.

On a write-enabled cube, you can use cube roles to grant read/write access to users and groups of users, and to limit access to specific cells or groups of cells in the cube. For more information about granting read/write access to a cube's cells, see Cell Security.

You can browse or delete writeback data. You can also convert writeback data to a partition.

The following table lists links to topics that provide more detailed information.

Topic	Description
Write-Enabling a Cube	Describes using the **Write-Enable** dialog box to write-enable a cube.
Browsing Writeback Data	Describes using the **Browse Data** dialog box to browse data for a write-enabled cube.
Deleting Writeback Data and Write-Disabling a Cube	Identifies considerations when deleting writeback data for a cube.
Converting Writeback Data to a Partition	Identifies considerations when converting writeback data to a partition.

Note Converting to a partition is available only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

See Also

Write-Enabled Cubes

Write-Enabling a Cube

To write-enable a cube, use the **Write-Enable** dialog box, which is accessed by right-clicking a cube from the Analysis Manager tree pane.

To write-enable a cube

Browsing Writeback Data

You can browse the contents of a cube's writeback table in the **Browse Data** dialog box, which is accessed by right-clicking a write-enabled cube from the Analysis Manager tree pane.

To browse writeback data for a cube

Deleting Writeback Data and Write-Disabling a Cube

You can delete the contents of a cube's writeback table by using the **Delete Writeback Data** dialog box, which is accessed by right-clicking a write-enabled cube from the Analysis Manager tree pane.

IMPORTANT When you delete the writeback data for a cube, you also write-disable the cube. All Unrestricted Read/Write policies and read/write permissions for the cube's cells are disabled, and end users will not be able to change displayed cube data. (End users with disabled Unrestricted Read/Write policies or disabled read/write permissions will still be able to browse the cube.) Read and read contingent permissions are not affected.

To delete writeback data for a cube and write-disable it

Converting Writeback Data to a Partition

You can convert the data in a cube's writeback table to a partition.

CAUTION Incorrect use of partitions can result in inaccurate cube data. For more information, see Managing Partitions.

When you convert the writeback data for a cube to a partition, you also write-disable the cube. All Unrestricted Read/Write policies and read/write permissions for the cube's cells are disabled, and end users will not be able to change displayed cube data. (End users with disabled Unrestricted Read/Write policies or disabled read/write permissions will still be able to browse the cube.) Read and read contingent permissions are not affected.

This procedure causes the writeback table to become the new partition's fact table.

Note Converting to a partition is available only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

To convert writeback data to a partition, use the **Convert to Partition** dialog box, which is accessed by right-clicking a write-enabled cube from the Analysis Manager tree pane.

To convert a cube's writeback data to a partition and write-disable the cube

Specifying Drillthrough Options

Drillthrough is an operation in which an end user selects a single cell from a regular, virtual, or linked cube and retrieves a result set from the source data for that cell in order to get more detailed information. For end users to drill through, their client applications must support this capability.

By default, a drillthrough result set is derived from only the table rows that were evaluated to calculate the value of the selected cube cell. For example, in a three-dimension cube, if the selected cube cell is the intersection of 1998 and Mexico, and the displayed dataset is sliced by all Customers and the measure Sales, the result set from drillthrough is generated from only:

- Fact table rows that are for both 1998 and Mexico.
- **Time** dimension table rows that store 1998.
- **Location** dimension table rows that store Mexico.
- All rows in the **Customers** dimension table.

The result set is summarized using the joins in the cube's schema.

Note Drillthrough is not supported for cells that have values based on expressions such as calculated member cells or ones that are associated with custom member formulas. This includes cells that display values for the level members of a mining model dimension.

Drillthrough permissions are granted through membership in cube roles. The only end users who can drill through are those in cube roles that have been granted this ability. You can grant this ability in Cube Role Manager or through programming with Decision Support Objects (DSO). For information about setting drillthrough options using DSO, see AllowDrillThrough (MDStore Interface).

To enable or disable drillthrough for a regular cube, open Cube Editor. On the

Tools menu, click **Drillthrough Options**. Then, in the **Drillthrough Options** dialog box, select or clear the **Enable drillthrough** check box. You can choose, on the **Columns** tab, which columns to display on drillthrough.

To enable or disable drillthrough for a virtual cube, open Virtual Cube Editor. In the tree pane, select the virtual cube icon, and then, in the properties pane, set the **Enable Drillthrough** property to either **True** or **False**. For linked cubes, enable or disable drillthrough by changing the settings of the source cube or cubes on the publishing server.

If drillthrough is enabled for a cube, you can test drillthrough while you browse the cube's data in Cube Browser, Cube Editor, or Virtual Cube Editor. To do this, right-click a cube cell, and then click **Drill Through**.

Specifying Drillthrough Options for Regular Cubes

You can enable drillthrough for individual regular cubes. For each cube, you can select the columns and tables that are included in the result set returned by a drillthrough operation. These columns can be from any table in the cube's data source, including those that are not part of the cube's schema. If drillthrough is enabled, at least one column must be selected.

Because a drillthrough operation can consume an extremely large amount of resources, enable drillthrough only after careful consideration. Grant cube roles the ability to drill through only after testing drillthrough with the cube.

One way to limit the resources used by a drillthrough is to specify a drillthrough filter. This filter is a WHERE clause expression added to the SQL SELECT statement that generates the result set. The filter can limit the number of rows in the result set.

If you want to enable drillthrough for a multiple-partition cube, before you begin the following procedure, make sure that the columns that you want to display in the result set exist in the tables for all of the cube's partitions. (The **Drillthrough Options** dialog box, where drillthrough is enabled, displays the column names and table names for only the cube's default partition.) Qualifying fact table names do not need to be the same in all partitions. If necessary, Microsoft® SQL Server™ 2000 Analysis Services automatically changes the query to reference the appropriate fact table name for each partition.

Note If a cube contains multiple partitions, drillthrough within the cube returns multiple result sets, one per partition. A client application might attempt to merge these result sets before presentation to the end user, thus yielding unexpected results.

To specify drillthrough options for a regular cube, use Cube Editor and the **Drillthrough Options** dialog box.

To specify drillthrough options for a regular cube

Analysis Services

Updating Cubes and Dimensions

The following topics contain information about updating cubes and dimensions in Microsoft® SQL Server™ 2000 Analysis Services.

Topic	**Description**
Updating and Refreshing Cube Data	Discusses the type of processing you must use depending on the type of change being made.
Updating and Rebuilding Dimensions	Contains information about how to incorporate changes to shared and private dimensions.

Updating and Refreshing Cube Data

Many of the changes you make within Analysis Manager and all of the changes to a cube's source data require the cube to be processed in order for the changes to be reflected in the cube's data.

You can process a cube in the following ways:

- Incremental update

  Adds new data to a partition in the cube and updates aggregations. This method does not process changes to a cube's structure (measures, dimensions, and so on) or changes to its existing source data. An incremental update creates a temporary partition from the new data and merges it into an existing partition.

- Refresh data

  Clears and reloads a cube's data and recalculates its aggregations. Use this method if the cube's source data has changed but its structure has not.

- Full process

  Completely restructures a cube based on its current definition and then recalculates its data.

For more information about the kinds of changes processed by the preceding methods, see Processing Cubes.

Because an incremental update creates a temporary partition from the new data and merges it into an existing partition, it is necessary to understand the special considerations that apply to partitions before performing an incremental update. For more information, see Incremental Updates and Partitions.

Referential integrity of the data warehouse is not verified by Microsoft® SQL Server™ 2000 Analysis Services. So, for example, if the cube's (or one of its partition's) fact table contains foreign key values that are not present in a joined dimension table's primary key column, the rows containing those values are not processed. In this case, processing does not produce an error message, but the cube contains incomplete and therefore inaccurate data.

To incrementally update a cube, use the **Process a Cube** dialog box, the Incremental Update Wizard, and the **Process** dialog box.

To incrementally update a cube

Updating and Rebuilding Dimensions

Different procedures are used to update or rebuild a dimension, depending on whether it is shared or private. For more information about the differences between shared and private dimensions, see Introduction to Dimensions.

Topic	Description
Updating and Rebuilding Shared Dimensions	Contains information about what type of processing to use depending on how the data in a shared dimension has changed.
Updating and Rebuilding Private Dimensions	Contains information about what type of processing to use depending on how the data in the private dimension has changed.

See Also

Dimension Processing

Updating and Rebuilding Shared Dimensions

After you update a shared dimension or its table, you must process it. You can process a shared dimension in the following ways:

- Incremental update

  Processes the addition of members (that is, rows) to the dimension table. This method does not process changes to a dimension's structure or relationships among members.

- Rebuild the dimension structure

  Processes changes to a dimension's structure such as adding, deleting, or moving a level. This method also processes changes to member relationships such as moving a member from one parent member to another.

Exceptions are changing dimensions, which do not always require processing by the second method after changes to the dimension's structure or changes to member relationships. Changing dimensions include virtual, parent-child, and relational OLAP (ROLAP) dimensions. For more information, see Changing Dimensions.

Important When a shared dimension is processed with the **Rebuild the dimension structure** option, all cubes that incorporate the shared dimension immediately become unavailable to users and must be processed before they can be used again.

Caution If a shared dimension's structure is updated and saved but not processed, it will automatically be processed when any cube incorporating the dimension is processed. At that time any other cubes that incorporate the dimension immediately become unavailable to users and must be processed before they can be used again.

To incrementally update a shared dimension or rebuild a shared dimension's structure, use the **Process a Dimension** and **Process** dialog boxes.

To incrementally update a shared dimension

Updating and Rebuilding Private Dimensions

After you update a private dimension or its table, you must process it. To process a private dimension, process the cube in which it is included. The type of cube processing that is required depends on the type of change made to the dimension. One of the following cube processing options is appropriate:

- **Refresh data** option

  Processes the addition of members (that is, rows) to the dimension table. This method does not process changes to a dimension's structure or relationships among members or deletion of members.

- **Full process** option

  Processes changes to a dimension's structure such as adding, deleting, or moving a level. This method also processes changes to member relationships such as moving a member from one parent member to another.

Exceptions are changing dimensions, whose cubes do not always require processing with the **Full process** option after changes to the dimension's structure or changes to member relationships. Changing dimensions include virtual, parent-child, and relational OLAP (ROLAP) dimensions. For more information, see Changing Dimensions.

For information about processing a cube by using the **Refresh data** option, see Updating and Refreshing Cube Data. For information about processing a cube by using the **Full process** option, see Processing Cubes.

Analysis Services

Building and Using Data Mining Models

Data mining, an exciting feature introduced as part of Microsoft® SQL Server™ 2000 Analysis Services, provides new tools for decision analysis by discovering patterns and rules in data and using them for predictive analysis, using industry standard data mining algorithms.

The primary mechanism for data mining is the data mining model, an abstract object that stores data mining information in a series of schema rowsets. Data mining models are easily accessible with a variety of tools. You can use the Mining Model Wizard to create data mining models, and you can use Data Mining Model Browser to display data mining model content in a graphical format.

You can also create, train, and use data mining models programmatically using OLE DB for Data Mining, an extension to the OLE DB specification that supports data mining functionality.

Data mining models can be used to perform sophisticated decision analysis on large amounts of data, whether relational or OLAP, and with a variety of algorithms. The following topics help you through the steps needed to create, train, and apply a data mining model, as well as view the content of a trained data mining model. Advanced operations, such as using mining model roles to provide security for a data mining model, are also covered.

Topic	Description
Creating Data Mining Models	Explains the use of the Mining Model Wizard for the creation of relational and OLAP data mining models.
Editing Data Mining Models	Explains the use of Mining Model Editor for editing relational and OLAP data mining models.
Training Data Mining Models	Details the process of training a data mining model.
Viewing Data Mining Models	Explains the use of Data Mining Model Browser and Dependency

	Network Browser for viewing and editing data mining model content.
Advanced Data Mining Model Operations	Covers the use of roles with data mining.

Creating Data Mining Models

Use the Mining Model Wizard to create new data mining models. The Mining Model Wizard takes you through several steps needed to establish the model type, build the model the case set will work with, and choose the data mining technique the model will use to construct a new data mining model.

There are two types of data mining models used, based on the type of case set data to be processed. A relational data mining model is designed to process traditional relational database tables, while an OLAP data model is designed to process OLAP data stored in the form of cubes. The creation of each type of mining model is covered in its own topic.

Data mining models share the same schemas and basic structures, regardless of whether they are based on relational or OLAP data. The most important element in the construction of a data mining model is the case. The case is a set of data mining columns used to define the information that the model will use to identify and study, and for which the model will provide prediction data.

Data mining columns vary in content type: Key columns are used to identify a specific case, input columns are used to provide information that the data mining provider can use to analyze a case, and predictive columns provide meaningful results, in the form of histogram data, from the analyzed case.

See Also

Creating OLAP Data Mining Models

Creating Relational Data Mining Models

Data Mining Columns

Data Mining Models

Creating Relational Data Mining Models

The Mining Model Wizard is used to create a relational data mining model. For more information about the Mining Model Wizard and the steps needed to create a relational data mining model, see Mining Model Wizard.

The Mining Model Wizard uses the case tables, data mining technique, case key column, input columns, and predictable columns you provide to build and optionally process a mining model.

The case tables contain the columns needed to establish a case set for a data mining model. In the case of a single case table, a single relational database table contains all of the information needed. With multiple case tables, however, the information may be distributed across several tables and joins between relational database tables are needed to establish the case set.

When you select a data mining technique, you choose a data mining algorithm to be used with the relational data mining model. By doing this, you implicitly select a data mining algorithm provider. The data mining algorithm provider supplies the data mining algorithms and dictates the model structure for the data mining model.

You also select input and prediction columns when you use the wizard to create a relational mining model. Input columns are used by the data mining provider to contain training data. Selection columns are used to provide predictive analysis results when querying a data mining model.

Prerequisites

Before you build a relational data mining model, a relational database containing a table structure for training data must exist, so that the Mining Model Wizard can use the table structure to define data mining columns. The Mining Model Wizard can also train the data mining model. Relational data mining models cannot be created from OLAP data sources.

See Also

Data Mining Algorithms

Data Mining Columns

Data Mining Models

Relational Model Steps (Mining Model Wizard)

Creating OLAP Data Mining Models

The process of creating an OLAP data mining model is similar to the process of creating a relational data mining model. To create an OLAP mining model, use the Mining Model Wizard. The steps needed to create an OLAP mining model are nearly identical to those needed to create a relational mining model.

The Mining Model Wizard uses the source cube, data mining technique, case dimension and level, predicted entity, and training data to create an OLAP data mining model.

The source cube provides the Mining Model Wizard with the information needed to both create a case set for the data mining model. Because a cube may contain many groups of information, the model uses a dimension and level that you choose from the cube to establish key columns for the case set.

When you select the data mining technique, you also select a data mining provider. The data mining provider provides the data mining algorithms and model structure for the data mining model.

The case dimension and level provide a specific orientation for the data mining model into the cube for creating a case set.

The predicted entity can be one of the following entities:

- A measure of the source cube
- A member property of the case dimension and level, selected earlier in the Mining Model Wizard
- Members of another dimension in the cube

This provides flexibility in dealing with the potentially complex process of predictive analysis on OLAP data.

Training data, in the form of dimensions, levels, member properties and measures, is used to process the OLAP data mining model and further define the data mining column structure for the case set.

Optionally, the wizard can create a new dimension for the source cube or a virtual cube based on the source cube. This enables users to query the data mining data just as they would query OLAP data.

Prerequisites

The columns in an OLAP data mining column are constructed from visible levels in the OLAP cube on which the mining model is based. To create an OLAP data mining model, the Mining Model Wizard requires a cube whose dimensions contain at least one visible level.

OLAP data mining models cannot be created from relational data sources, and virtual dimensions created in Microsoft® SQL Server™ 7.0 OLAP Services cannot be used in OLAP mining models. Also, an OLAP mining model cannot be based on a virtual cube that contains a data mining dimension.

See Also

Mining Model Wizard

Data Mining Algorithms

Data Mining Columns

Data Mining Models

OLAP Model Steps (Mining Model Wizard)

Editing Data Mining Models

The process of editing data mining models depends upon the type of mining model; relational and OLAP data mining models each have an editor, designed to meet the specialized needs of each model.

See Also

Editing OLAP Data Mining Models

Editing Relational Data Mining Models

Editing Relational Data Mining Models

You can use Relational Mining Model Editor to edit the structure of relational data mining models. Relational Mining Model Editor can also be used to process a data mining model and view the resulting content.

Relational Mining Model Editor enables you to change basic properties, such as the data mining algorithm, of the data mining model. It shows the data mining model columns, including key, input, and predictive columns, and it allows you to edit the column properties individually.

Relational Mining Model Editor also displays the table schema used to construct the case set in the Mining Model Wizard, showing both case and supporting tables.

For trained relational mining models, Relational Mining Model Editor can display the data mining model content graphically using Data Mining Model Browser.

See Also

Data Mining Model Browser

Relational Mining Model Editor

Editing OLAP Data Mining Models

You can use OLAP Mining Model Editor to edit the structure of OLAP data mining models. And, similar to Relational Mining Model Editor, OLAP Mining Model Editor can process a data mining model and display the resulting content.

OLAP Mining Model Editor enables you to change basic properties, such as the data mining algorithm, of the data mining model. The editor also enables you to change the properties of the dimensions, levels, and measures that compose the case set for the data mining model.

If the OLAP data mining model has been trained, OLAP Mining Model Editor also displays the data mining model content graphically using Data Mining Model Browser.

See Also

Data Mining Model Browser

OLAP Mining Model Editor

Training Data Mining Models

In order for data mining models to provide predictive results, they first must work with known data in a process known as training. During this process, data is inserted into the untrained data mining model. The process of inserting the data does not save the training data into the data mining model; rather, the data mining model analyzes the training data, looking for rules and patterns that can be used later to determine the histogram values for predictive columns, and then it stores the statistical information as data mining model content.

Training is done by processing the data mining model in the Mining Model Wizard, in the mining model editors, and from Analysis Manager.

The training process is similar for both relational and OLAP mining models. In the Mining Model Wizard, the source tables or cube used to construct the model are assumed to contain training data and are used to supply the data mining model. In the Mining Model Editor, the case and association tables for a relational data mining model expected to supply the training data are displayed as part of the model. This is not so with the OLAP mining model, because all of the dimensions, levels, and measures of the source cube are duplicated as part of the structure of the OLAP mining model, even if they are not being employed as an active part of the mining model. A relational data mining model incorporates only those data mining columns that will be used by the mining model into its structure.

Processing the data mining model can be performed in one of the following ways.

Refresh

> Clears the data mining model content and retrains the model from the training data. It is best used when the model structure has not changed, but the model needs to be completely retrained from a new set of training data. End users can continue to query a data mining model during a refresh process; after the refresh process completes, the users have access to the refreshed data without having to reconnect.

Full Process

Completely removes and rebuilds the data mining model and trains the newly constructed model from the training data. It is required for data mining models whose structure has changed and for models that have not yet been trained. End users working with the data mining model must reconnect to the server after this process completes in order to continue to work with it.

After the mining model is processed, the information about the patterns and rules discovered in the training data are stored as data mining model content, along with the distribution information of the case data as well.

See Also

Mining Model Wizard

Analysis Manager

OLAP Mining Model Editor

Relational Mining Model Editor

Viewing Data Mining Models

The easiest way to view the contents of a trained data mining model is to use Data Mining Model Browser and Dependency Network Browser. These visualization tools display the complex contents of a data mining model, such as a decision tree, in an understandable graphical interface. The browsers enable you to visualize the data mining model content.

Visualization, the process of displaying complex data in a visual format that can be easily comprehended, can be difficult when it comes to data mining. Data Mining Model Browser simplifies the visualization of data mining model content, while Dependency Network Browser makes visualization of the complex relationships within a decision tree data mining model readily understandable.

By browsing the data mining model content, you can detect spurious relationships, which can suggest overfitting or other mining model problems, allowing you to modify and fully process the data mining model later using one of the mining model editors.

The following topics describe the viewers supplied with Microsoft® SQL Server™ 2000 Analysis Services in more detail.

Topic	Description
Viewing with Data Mining Model Browser	Describes how to use Data Mining Model Browser to study distribution information in detail for a data mining model.
Viewing with Dependency Network Browser	Details how to use Dependency Network Browser to understand complex dependency networks within decision tree data mining models.

See Also

OLAP Mining Model Editor

Relational Mining Model Editor

Data Mining Model Browser

Dependency Network Browser

Viewing with Data Mining Model Browser

Data Mining Model Browser allows you to view data mining content from the vantage point of a single attribute and its relationships. It shows the mining model content for each node that is influenced by a single attribute, as well as histogram data for each node. It displays the data mining model nodes used in the mining model, including the relationships between the nodes and the rules or attributes assigned to them, as an interconnected network of boxes. Each box represents a node in a single decision tree or a single cluster.

The nodes are color-coded to represent the data density of an attribute applicable to a selected node in relation to the total number of cases processed by the selected node. The color coding and selected attribute can be changed through the use of the tree color drop-down list on the legend pane.

The nodes are represented in ranking order of attribute factors, from left to right, in the content detail pane. The further down the tree a split is represented, the less influence the factor that caused the carries in the data mining model. Additionally, the attributes pane allows sorting of attributes by number of cases or probability of occurrence in the selected node, allowing you to better understand the relevance of a given attribute to a node.

The benefit of data mining model content visualization with Data Mining Model Browser is the understanding of the patterns and rules that encompass a case set, and the ability to fine tune these patterns and rules to better fit training data. For example, you can use the visualization capabilities of Data Mining Model Browser to eliminate a common problem in data mining called *overfitting*. Overfitting occurs when the data mining model starts constructing rules that are specific to single cases; the model starts attaching importance to unimportant patterns. For example, assume that there is a customer case set for a department store data mining model, which includes the last name of the customer as an attribute field. The data mining model might create a rule where a customer named Smith is most likely to purchase tools because a single customer named Smith purchased tools. This rule is based on a random pattern, which has no meaningful content. This rule is an example of overfitting; the correlation between the last name of a customer and the type of products purchased is meaningless. Overfitting occurs most often when attributes are added to a data mining model that do not supply meaningful content. In such cases, the model attempts to construct rules where none should exist.

The information shown in Data Mining Model Browser represents the statistical model of trends learned by the data mining model through the review of training data. As such, you will find it useful to review the attributes and node paths that define the knowledge gained by training a data mining model to better understand the general patterns and rules represented by the training data.

See Also

Data Mining Model Browser

OLAP Mining Model Editor

Relational Mining Model Editor

Dependency Network Browser

Viewing with Dependency Network Browser

Dependency Network Browser views data mining content for decision tree mining models from a vantage point different from that of Data Mining Model Browser. Whereas Data Mining Model Browser allows you to view relationship and distribution information from the viewpoint of a single attribute, Dependency Network Browser gives you the ability to view the data mining model from the viewpoint of all attributes by relationship information alone, providing a wider perspective on the entire data mining model.

Consider the following example. You have constructed a data mining model based on product purchases for customers using store credit cards. The customer demographic information and product order information is quite rich; numerous input and predictable columns have been supplied for the data mining model. Using Dependency Network Browser, you can get an immediate sense of how well customer demographic information predicts other customer demographic information, for example, and how the same customer demographic information predicts product order information. Dependency Network Browser shows the relationships contained in every tree in the data mining model, information that would be difficult to extrapolate by viewing each tree in isolation.

After an attribute is isolated, Data Mining Model Browser allows you to view the details and distribution information for the relationships of the selected attribute.

Dependency Network Browser displays all of the attributes in the data mining model as nodes, with arrows indicating prediction links between nodes. For example, an arrow from an Exercise node to a Heart Rate node indicates that the Exercise attribute predicts the Heart Rate attribute. The strength of the prediction link can be viewed by the use of a slider on the left side of the window. If the slider is set to **All links**, all prediction links are shown, no matter how little they affect the related nodes. If the slider is set to **Strongest links**, only the strongest prediction links are displayed.

The nodes are color-coded to represent the selected node and the direction of predictability of related nodes. The nodes can also be moved by clicking and dragging to improve the view of relationships, or the **Improve Layout** button can be used to automatically distribute and resize nodes for better viewing.

See Also

Dependency Network Browser

OLAP Mining Model Editor

Relational Mining Model Editor

Data Mining Model Browser

Advanced Data Mining Model Operations

The advanced task in this topic allows you to improve the performance and accuracy of a data mining model and apply security features to mining models by using database roles.

Topic	Description
Using Roles in Data Mining Models	Explains the use of database roles as a method of securing data mining models.

Using Roles in Data Mining Models

Data mining models, like other objects managed by Microsoft® SQL Server™ 2000 Analysis Services, can have security roles assigned to them to restrict access to the model and to its prediction capabilities to specific users and groups.

As data mining models are linked to databases in Analysis Manager, database roles are employed to grant or deny access to a data mining model linked to a database, similar to the use of a database role with a cube. Mining Model Role Manager handles the task of creating, editing, and maintaining mining model roles.

CAUTION Because database roles can be shared with other cubes and data mining models, a change made to the definition of a role, such as the name or the user membership, affects other cubes or data mining models that use the role.

See Also

Database Roles

Mining Model Role Manager

Analysis Services

Archiving, Restoring, and Copying Data

Microsoft® SQL Server™ 2000 Analysis Services provides functionality to help you copy, move, or back up objects in Analysis Manager. You can use Analysis Manager or the **msmdarch** command to archive and restore Analysis Services databases. Each database is archived to one or more .cab files, which you can reserve for restoration requirements or migrate to other server computers. A .cab file is called an archive file. You can also copy and paste various objects in Analysis Manager.

The following topics contain information about archiving, restoring, and copying data in Microsoft SQL Server 2000 Analysis Services.

Topic	Description
Archiving and Restoring Databases	Describes how databases are archived and restored in Analysis Services.
Copying and Pasting Objects	Contains information about duplicating or moving objects in Analysis Manager.

Archiving and Restoring Databases

An archive file for a Microsoft® SQL Server™ 2000 Analysis Services database contains the contents of the directory named the same as the database. For example, the archive file for the **FoodMart 2000** sample database stores the contents of the FoodMart 2000 directory. The default path of this directory is:

```
C:\Program Files\Microsoft Analysis Services\Data\FoodMart 2000
```

All Analysis Services databases can be found in the Data directory. The path leading to the Data directory can be changed. To determine the current path, in Analysis Manager, right-click the server that contains the database, click **Properties**, and then see the **Data folder** box.

Important The subdirectories of the Data directory store security files that control end users' access to objects on the Analysis server. These files are included in the archive files. For this reason, archive files must be secured against unauthorized access.

The archive file also stores meta data for the database and its objects. The appropriate records from the Analysis Services repository are included in the archive file. By default the Analysis Services repository is:

```
C:\Program Files\Microsoft Analysis Services\Bin\msmdrep.mdb
```

However, the path of the repository can be changed at installation, and the repository can be migrated to a SQL Server database.

Caution When you restore a database, its file set (in the Data directory and its subdirectories) and its meta data are returned to their states at the time the archive file was created. Files that were created since then in these directories are deleted. Changes and additions since then to Analysis Services repository records for the database and its objects are removed.

When you restore a database:

- The files in the selected archive file are restored to the appropriate directories in the Data directory. If these directories already contain files with the same names as files in the archive file, the existing files are overwritten by the files in the archive file. If these directories contain files with different names, including new files that were created since the archive file was created, these files are deleted.

- The records in the Analysis Services repository that are associated with the database and its objects are replaced by the Analysis Services repository records in the selected archive file.

Sometimes restoration of a database containing relational OLAP (ROLAP) partitions copies rather than replaces the original, archived partitions. When this happens, the original ROLAP partitions and their copies will use the same aggregation tables. This situation may cause overwrite conflicts in the tables. For example, if you change and process an original partition, the aggregation tables may no longer be valid for the copy of that partition. To avoid this problem, specify a unique aggregation prefix for each ROLAP partition immediately after restoration, and then process the partitions whose aggregation prefixes you changed. This action creates different aggregation tables for each partition. To access the aggregation prefix for a partition, in the Analysis Manager tree pane, right-click the partition, click **Edit**, advance to the final step of the Partition Wizard, and then click **Advanced**.

Caution Archiving a database does not archive writeback tables, source data, or aggregations for ROLAP partitions in that database. Writeback tables, source data, and aggregations for ROLAP partitions are required for ongoing, correct operation of Analysis Services. You must archive or back up this data with the backup software you ordinarily use because it is not contained in archive files created by Analysis Services.

Writeback tables are not stored in the archive file. Therefore, if you restore a write-enabled cube and its writeback table is not available, the cube must be processed before it can be used. After processing, the effects of the writeback data are absent from the cube's data. For more information about writeback data and write-enabled cubes, see Write-Enabled Cubes.

Data in remote partitions is not archived or restored with a database. After you restore a database that has a remote partition, you must process the remote partition. For more information about remote partitions, see Remote Partitions.

To process a partition, see How to process a partition.

The contents of the archive file vary according to the storage types of the partitions in the database. The following table indicates these variations.

Storage type	Source data is contained in archive file	Copy of source data usable by partition is contained in archive file	Aggregations are contained in archive file
MOLAP	No	Yes	Yes
ROLAP	No	No	No
HOLAP	No	No	Yes

Restored MOLAP partitions are usable even if their source data (that is, the tables in the data source used by the partition) is lost or unavailable. However, a restored MOLAP partition and its parent cube cannot be updated if the source data for the partition is lost permanently. Restored ROLAP and HOLAP partitions rely on the availability of their source data for correct operation. Restored ROLAP partitions also rely on the availability of their aggregation tables or indexed views, which are stored with the source data.

Archiving an Analysis Services Database

You can archive a Microsoft® SQL Server™ 2000 Analysis Services database by using Analysis Manager or the **msmdarch** command.

To archive an Analysis Services database using Analysis Manager, use the **Archive Database** and **Archive Database Progress** dialog boxes.

To archive an Analysis Services database using Analysis Manager

Restoring an Analysis Services Database

You can restore a Microsoft® SQL Server™ 2000 Analysis Services database by using Analysis Manager or the **msmdarch** command.

To restore an Analysis Services database with the Analysis Manager, use the **Open Archive File**, **Restore Database**, and **Restore Database Progress** dialog boxes. If you are restoring a database on a remote server, you must also use the **Remote Server Data Directory** dialog box.

To restore an Analysis Services database using Analysis Manager

Copying and Pasting Objects

You can use Analysis Manager to copy and paste many types of objects used in Microsoft® SQL Server™ 2000 Analysis Services. The **Copy** and **Paste** commands are provided on the following menus:

- The shortcut (that is, right-click) menus in the Analysis Manager tree pane
- The Analysis Manager **Action** menu

You can copy and paste objects within a database, between databases, and between servers.

To copy and paste an object

Analysis Services

Analyzing and Optimizing Performance

Microsoft® SQL Server™ 2000 Analysis Services provides tools that you can use to analyze and optimize the performance of your Analysis Services installation.

Topic	Description
Analyzing Usage Patterns	Describes the Usage Analysis Wizard. You can use this wizard to generate performance reports.
Optimizing Performance Based on Usage	Describes the Usage-Based Optimization Wizard. You can use this wizard to optimize aggregation design based on past queries.
Optimizing the Data Warehouse Database for Analysis Services Performance	Discusses techniques you can use when designing your data warehouse that will improve the performance of Analysis Services.
Optimizing Cube Schemas	Discusses the **Optimize Schema** option for cubes and how it can substantially reduce cube processing time.
Monitoring Analysis Services Performance	Details performance objects, counters, and instances you can use with Microsoft Windows 2000® System Monitor (known as Performance Monitor in Windows NT® 4.0) to monitor the performance of Analysis servers.

Analyzing Usage Patterns

To produce an on-screen report of a cube's query patterns quickly, use the Usage Analysis Wizard. Report content is derived from the query log, which, by default, records every tenth query. You can adjust the content of the query log by changing this interval, stopping logging altogether, or clearing the log to restart logging. For more information, see Logging Tab (Properties Dialog Box).

You can tailor report content by using criteria to select the log entries you want to include in the report. The following reports are available.

Report name	Displays
Query Run-Time Table	The run time of queries, ordered from the longest to the shortest run time.
Query Frequency Table	The frequency of queries, ordered from the most to the least frequent.
Active User Table	Users and the number of queries they have sent, ordered from the most to least queries sent per user.
Query Response Graph	The response time for all queries.
Query By Hour Graph	The total number of queries processed, grouped by hour.
Query By Date Graph	The total number of queries sent, grouped by date.

To start the Usage Analysis Wizard

Optimizing Performance Based on Usage

To optimize partition performance based on patterns of logged queries, use the Usage-Based Optimization Wizard, which takes you through steps to specify options for optimization. For more information, see Usage-Based Optimization Wizard.

You can optimize some or all of the partitions for cubes that contain multiple partitions, but each partition must be individually optimized. If a partition contains existing aggregations, you can either add the new aggregations to the existing ones or replace them. You can also change the storage mode of the partition you are optimizing. You must process a partition after optimizing its aggregations, and the final step of the wizard allows you to process immediately or defer processing.

Important Do not optimize partition aggregation designs if you might want to merge partitions in the future. You cannot merge partitions unless their aggregation designs are identical. For more information, see Partitions and Merging Partitions.

The usage-based optimization process designs aggregations tailored for query patterns recorded in the query log, which by default records every tenth query. You can adjust the content of the query log by changing this interval, stopping logging altogether, or clearing the log to restart logging. One approach to optimizing performance based on usage is to create partitions with zero aggregations, adjust query logging to log every query for a period of time to capture typical usage patterns, and then use the wizard to design aggregations appropriate to the usage. For more information, see Logging Tab (Properties Dialog Box).

To start the Usage-Based Optimization Wizard

Optimizing the Data Warehouse Database for Analysis Services Performance

The design and performance of the data warehouse database significantly affect the performance of Microsoft® SQL Server™ 2000 Analysis Services.

Analysis Services creates multidimensional presentations of data warehouse data by reading and organizing the data into multidimensional objects such as dimensions and cubes. The Analysis server uses the database relational engine to access the data warehouse database when creating and processing dimensions and cubes. Therefore, the data warehouse schema design and relational database performance have a significant effect on the ease of designing cubes and on the performance of processing cubes.

Cube Storage Modes

A cube's storage mode significantly influences the degree to which the data warehouse schema and relational database performance affect cube query performance.

- With multidimensional OLAP (MOLAP) storage, cube query performance does not depend on relational database performance because all of the cube's data is contained in the multidimensional structure used by the Analysis server.

- With hybrid OLAP (HOLAP) storage, only cube queries that require retrieval of facts from the fact table will require the Analysis server to query the relational database. Such queries are affected by database performance, but queries that do not access the fact table are unaffected.

- With relational OLAP (ROLAP) storage, all cube queries are affected by the database performance because the Analysis server must retrieve fact and aggregated data from the relational database. If dimensions also use ROLAP storage, cube query performance will depend to a large extent on the performance of the relational database. Caching in the Analysis server affects cube performance in this situation, but only to the extent that queries can be at least partially resolved from cached information.

Dimensional Modeling

The design of the data warehouse database schema should incorporate the principles of dimensional modeling so the dimension tables and fact tables represent the business data and the way clients will view and query the data. Most dimensional modeling results in a star or snowflake database schema. Such schemas facilitate cube design and reduce the number of multiple-table joins when querying the database to process dimensions and cubes. If a snowflake schema is needed, minimize the number of dimension tables beyond the first level from the fact table, and minimize the depth of the snowflake dimensions. Fact tables usually hold the vast majority of data in the data warehouse, sometimes containing hundreds of millions of rows. Fact tables should be carefully designed to eliminate duplicated data and to minimize the length of the rows.

Cube Processing

Relational database optimization techniques that improve the speed of reading the data will improve dimension and cube processing performance in Analysis Services. One of the most important optimization techniques is to design and use effective indexes on the fact and dimension tables to facilitate performance of the joins and queries Analysis Services issues when processing dimensions and cubes.

SQL Server 2000 offers a number of options and suggestions for optimizing logical and physical relational database design and query performance. Many of these techniques apply to data warehouse databases, as well as to transaction processing databases. For more information, see Optimizing Database Performance Overview.

Common Data Warehouse Techniques

Many common data warehouse design and optimization techniques apply regardless of the tools that are used to present data to clients. However, some optimization techniques that are appropriate when SQL queries are used as the presentation tool may not be necessary or appropriate if the tool is Analysis Services. In general, the granularity of the fact table should not be reduced by summarization; the number of fact table rows affects the time to process cubes but has little effect on OLAP client query response performance. Analysis Services uses sophisticated algorithms to create aggregation tables of summary data, so these need not be created in the database schema. Special bridge tables for parent-child dimensions such as organizational hierarchies are not necessary; Analysis Services can use self-referential dimension tables in their native form.

Record Size and Data Types

The size of a record affects Analysis Services performance in all areas, including cube size, processing time, server memory usage, server to client data transfer time, and client memory usage. Fact tables typically contain the vast majority of data in the data warehouse. Fact table records should be kept as short as possible and include only fields for measures and indexed key columns. Measure fields should use the smallest data type consistent with the measure data, but be sure the data type is large enough to contain summarized values to prevent overflow when aggregations are calculated. In a fact table containing millions of rows, a saving of even two bytes per record can amount to a significant reduction in table size and the time required to process the table when creating cubes.

Data Warehouse Updates

Because Analysis Services reads data warehouse data and stores it in multidimensional structures, the frequency of changes to the data warehouse data affects how often OLAP cubes and dimensions have to be reprocessed. A data warehouse update strategy should be designed to take the need for reprocessing OLAP objects into account.

Optimizing Cube Schemas

In many situations Microsoft® SQL Server™ 2000 Analysis Services can optimize a cube's schema to significantly reduce cube processing time by eliminating joins between dimension tables and fact tables.

During dimension processing, the Analysis server creates an internal representation of the dimension data and hierarchy. When processing a cube, the dimension member keys identified in the member key column property are used to access the information in the internal representation of the processed dimension. Under certain conditions, the dimension member's foreign key in the fact table can be used for this lookup, thereby eliminating the need to join the dimension table to the fact table in the database query. This significantly reduces the complexity of the query, the amount of data accessed in the relational database, and network traffic between the Analysis server and the relational database.

To take advantage of cube schema optimization, when you design a cube in Cube Editor, click the **Optimize Schema** command on the **Tools** menu. Analysis Services then modifies the schema to eliminate joins between the fact table and dimension tables, where possible. Certain conditions must be met for Analysis Services to eliminate a join between a dimension and the fact table. These are:

- The dimension must be a shared dimension, and must have been processed before you optimize the cube schema.

- The member key column for the lowest level of the dimension must contain the keys that relate the fact table and the dimension table. This must be the only key necessary to relate the fact table to the dimension table.

- The keys in the member key column for the lowest level of the dimension must be unique.

- The lowest level of the dimension must be represented in the cube, that is, the level's **Disabled** property must be set to **No**. The level can be hidden.

If these conditions are met, and the cube's schema is optimized using the **Optimize Schema** option, the Analysis server ignores the dimension table in the database when processing the cube. If these conditions are met for all dimensions in the cube, the Analysis server needs to read only the fact table to process the cube. Processing time reductions often can be substantial when this optimization technique is used.

Cube schema optimization applies to all partitions of the cube whether the partitions are processed independently or as a group.

Note You should not optimize a cube's schema if you depend on inner joins between the fact table and dimension tables to exclude fact rows for the cube content. The entire fact table is read if all dimension table joins are removed by this optimization.

Because schema optimization can eliminate joins, a cube with an optimized schema may not display all available tables for use when specifying drillthrough options. You can join a table to the schema for drillthrough when specifying drillthrough options by adding the table and defining a SQL WHERE clause to establish the join. For more information, see Specifying Drillthrough Options.

Member Key Column

Analysis Services uses the **Member Key Column** property of the lowest level of a dimension to control cube schema optimization. During cube schema optimization, each dimension is evaluated to determine if it meets the conditions for optimization. If the dimension meets the required conditions, the **Member Key Column** property of the lowest level of the dimension is changed to refer to the foreign key in the fact table instead of the key in the dimension table. For example, before optimization the dimension level's **Member Key Column** is "Products"."SKU_ID", which is joined to the key, "Facts"."SKU_Key", in the fact table. After optimization, the **Member Key Column** property value is "Facts"."SKU_Key". This signals the Analysis server to use the key from the fact table during processing instead of issuing queries that join the dimension to the fact table in the relational database.

Example

A dimension for time contains the levels Year, Quarter, Month, and Day. There is a dimension member for each day in the dimension, and each day member has a unique key, which is specified as the member key column for the Day level. The **Member Keys Unique** property for the Day level is set to **True**.

The dimension's Day level member keys are used as foreign keys in the fact table to relate the dimension table to the fact table. No other keys are required to uniquely relate a fact table row to a row in the dimension table.

A cube is designed that uses this fact table and this time dimension. It is preferable that the cube contain summarized data at the Month level and above but not at the Day level. In the cube, the **Disabled** property for the dimension's Day level is set to **No**, so the level keys will be available for the cube processing optimization. The **Visible** property for the dimension's Day level is set to **False**, so the cube will not display data for the Day level.

When the **Optimize Schema** command on the **Tools** menu is selected, the cube schema is optimized. Then, when the cube is saved and processed, the SQL query issued by the Analysis server to read the fact table will not need to join or access the dimension table.

Modifying Cube Schema Optimization

You can remove the optimization for one or more dimensions in a cube by changing the **Member Key Column** property for the lowest level of each of the dimensions to refer to its original column in the dimension table. This will cause the Analysis server to issue a query that joins the dimension table to the fact table during processing.

Note A cube's schema optimization can be affected by adding or deleting dimensions, or by modifying dimension properties in the cube. You should check the cube's schema optimization or redo the optimization whenever you make such changes.

Unknown Dimension Member Error

This error indicates that a dimension member's key is not found in the internal representation of a dimension when processing a cube that contains the dimension. The cause can be either that a dimension has not been processed after new members were added, or that the dimension table does not contain a key that matches a key found in the fact table.

This error occurs regardless of whether a cube's schema has been optimized if new members are added to a dimension and related facts are added to the fact table but the dimension has not been processed. It makes no difference whether the member keys are read from the joined dimension (schema not optimized) or from the member foreign keys in the fact table (schema optimized). The internal representation of the dimension will not contain the new keys until the dimension has been processed.

There is one situation where this error is triggered if the cube's schema has been optimized, but is not be triggered if the schema has not been optimized. This condition occurs when a fact has been added to the fact table but no corresponding member exists in a dimension table. If the cube's schema has been optimized, the key for the new fact will be read from the fact table but not found in the internal representation of the dimension, even if the dimension has been processed. However, if the cube's schema has not been optimized, the query that joins the dimension table to the fact table causes any facts that do not have corresponding dimension members to be ignored and not read during processing, so the error is not triggered.

You can avoid these errors by maintaining referential integrity between dimension tables and fact tables, and by always processing a dimension after making changes to the dimension table and before processing cubes that use the dimension.

Monitoring Analysis Services Performance

System Monitor is a graphical tool in Microsoft® Windows® 2000 for measuring the performance of your own computer or other computers on a network. (In Microsoft Windows NT® 4.0, this tool is called Performance Monitor). You can use either System Monitor or Performance Monitor to do the tasks described in this section. For each computer, you can view the behavior of objects, such as processors, memory, cache, threads, and processes. Each of these objects has an associated set of counters that measure device usage, queue lengths, delays, and other indicators of throughput and internal congestion. The dynamic link library that provides the performance objects and counters for Microsoft® SQL Server™ 2000 Analysis Services, MSMDCTR80.DLL, is installed into the \Bin subdirectory of the Analysis Services directory.

Windows System Monitor provides charting, alerting, and reporting capabilities that reflect both current activity and ongoing logging. You can open, browse, and chart log files later as if they reflected current activity.

Installing and Uninstalling Analysis Services Performance Counters

To manually install the performance objects and counters supplied by Analysis Services, for use with the System Monitor, the following steps must be performed.

Installing Analysis Services Performance Counters

1. Add the named values in the following table to the following registry key:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Se
   ```

   The registry path is described in the following table.

Named value	Value
Library	The file path of the Msmdctr80.dll

	library
Open	OpenPerformanceData
Collect	CollectPerformanceData
Close	ClosePerformanceData

2. The information for the performance counters must be installed into the registry. Two files, Msmdctr.ini and Msmdctr.h, are included for this purpose. The two files are typically installed in the same location as the msmdctr80.dll library. To load this information into the registry, use the LODCTR.EXE program distributed with Windows NT 4.0 and Windows 2000. To install, execute the following command in a command prompt:
   ```
   LODCTR msmdctr.ini
   ```

Uninstalling Analysis Services Performance Counters

1. To uninstall the performance objects and counters, use the UNLODCTR.EXE program distributed with Windows NT 4.0 and Windows 2000 for this purpose. To uninstall, execute the following command in a command prompt:
   ```
   UNLODCTR MSSQLServerOLAPService
   ```

Analysis Services provides objects and counters that can be used to monitor activity in computers running Analysis Services. The topics in the following table of objects provide more detailed information about monitoring Analysis Services performance.

Analysis Services objects	**Description**
Analysis Server:Agg Cache Object	Collects statistical information about aggregation cache as related to Analysis Services
Analysis Server:Connection Object	Collects statistical information about connections as related to Analysis Services
Analysis Server:Last Query Object	Collects statistical information about

	the last Analysis Services query
Analysis Server:Locks Object	Collects statistical information about internal server latches and locks as related to Analysis Services
Analysis Server:Proc Object	Collects statistical information about processing data as related to Analysis Services
Analysis Server:Proc Aggs Object	Collects statistical information about the processing of aggregations in multidimensional OLAP (MOLAP) data files
Analysis Server:Proc Indexes Object	Collects statistical information about the processing of indexes for MOLAP data files
Analysis Server:Query Object	Collects statistical information about Analysis Services queries
Analysis Server:Query Dims Object	Collects statistical information about Analysis Services query of dimensions and meta data
Analysis Server:Startup Object	Collects statistical information about Analysis Services startup

See Also

Analyzing and Optimizing Performance

Analysis Server:Agg Cache Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Agg Cache** object. The **Agg Cache** object counters provide information about the aggregation cache.

Agg Cache object counters	Description
Bytes added/sec	Rate of bytes added to the cache.
Current bytes	Current number of bytes used by the aggregation cache.
Current entries	Current number of cache entries.
Direct hit ratio	Ratio of cache direct hits to cache lookups, for the period between obtaining counter values.
Direct hits/sec	Rate of cache direct hits. Queries were answered from an existing cache entry.
Evictions/sec	Rate of evictions from the cache. This is per partition per cube per database. Typically due to background cleaner.
Filter hit ratio	Ratio of cache filter hits to cache lookups, for the period between obtaining counter values.
Filter hits/sec	Rate of cache filter hits. Queries were answered by filtering an existing cache entry.
Inserts/sec	The rate of insertions into the cache. This is per partition per cube per database.
Lookups/sec	The rate of cache lookups.
Misses/sec	The rate of cache misses.
Total direct hits	Total count of direct cache hits. Queries were answered from existing cache entries.
Total filter hits	Total count of filter cache hits. Queries were answered by filtering existing cache entries.
Total evictions	Evictions from the cache. This is per partition per cube per database. Typically due to

	background cleaner.
Total inserts	Insertions into the cache. This is per partition per cube per database.
Total lookups	Total number of lookups into the cache. Note that each MDX query has zero or more server round trips, and each partition will be queried.
Total misses	Total count of cache misses.

See Also

Monitoring Analysis Services Performance

Analysis Server:Connection Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Connection** object. The **Connection** object counters provide information about server activity in general. They can be used to monitor the performance of user connections with Analysis Services.

Connection object counters	Description
Authentications/sec	Rate of user authentications.
Completions/sec	Rate of completed connections. This is derived by adding both successful and failed completions.
Current agents	Current number of agents on server.
Current authentications	Current number of authentications in progress.
Current connections	Current number of client connections established.
Current connections in progress	Current number of connections pending completion.
Current http connections	Current number of HTTP connections established.
Failures/sec	Rate of failures.
Requests/sec	Rate of requests. These are arrivals.
Successes/sec	Rate of connections successfully completed.
Total authentications	Total user authentications.
Total completions	Total connection completions.
Total failures	Total failed connection attempts.
Total requests	Total connection requests. These are arrivals.
Total successes	Total successful connections.

See Also

Monitoring Analysis Services Performance

Analysis Server:Last Query Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Last Query** object. The **Last Query** object counters provide information about processing on the last query.

Last Query object counters	Description
Answer from cache direct	Number of answers directly from cache. Note that this counter is per partition.
Answer from cache filtered	Number of partitions by filtering from other cache entries.
Answer from file	Number of partitions contributing to the answer from files.
Data avg bytes/read	Average bytes per data file per read operation.
Data avg bytes/row	Average bytes per row read during the last query.
Data avg rows/read	Average number of rows read per data file read operation during the last query.
Data bytes	Bytes read from the data file during the last query.
Data bytes total	Data file size in bytes during the last query. Total for all multidimensional OLAP (MOLAP) partitions examined.
Data reads	Number of logical I/O reads against the Data file during the last query.
DSN requested	Data Set Name requested as the query to the server. (Levels used for each dimension.)
DSN used	Data Set Name used to answer the query. (Levels used for each dimension.)
Index bytes	Number of bytes read from the Index file during the last query.
Index reads	Number of logical I/O reads against the Index

	file during the last query.
Map bytes	Bytes read from the Map file during the last query.
Map reads	Number of logical input/output (I/O) reads against the Map file during the last query.
Query num	Monotonically increasing query count. Might be useful for tools to check which query this information applies to.
Rows created	Number of rows created, which forms the final answer in the aggregation set. Because this is the result of aggregation this counter will usually be much smaller than the Rows read counter.
Rows filter excluded	Number of rows that a filter has excluded during the last query.
Rows filter included	Number of rows that a filter has included during the last query.
Rows filtered	Number of rows against which a filtering operation was applied during the last query.
Rows read	Number of rows read from a disk (facts or aggregations) or from the aggregation cache (in memory).
Time (ms)	Elapsed time in milliseconds.
Total bytes	Bytes read from all files.
Total reads	Number of logical I/O reads against all files processed during the last query.

See Also

Monitoring Analysis Services Performance

Analysis Server:Locks Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Locks** object. The **Locks** object counters provide information about latching and locking activity. They can be used to monitor the performance of latches and locks with Analysis Services.

Latch object counters	Description
Current latch waits	Current number of threads waiting for a latch. These are latch requests that could not be given immediate grants and are in a wait state.
Current lock waits	Current number of clients waiting for a lock.
Current locks	Current number of locked objects.
Latch waits/sec	Rate of latch requests that could not be granted immediately.
Lock denials/sec	Rate of lock denials.
Lock grants/sec	Number of lock grants per second.
Lock requests/sec	Number of lock requests per second.
Lock waits/sec	Number of lock waits per second. These are lock requests that could not be given immediate lock grants and are in a wait state.
Unlock requests/sec	Number of unlock requests per second.

See Also

Monitoring Analysis Services Performance

Analysis Server:Proc Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Proc** object. The **Proc** object counters provide information about processing data.

Proc object counters	Description
Current partitions	Current number of partitions being processed.
Current threads merging	Current number of threads merging data. Decremented when blocked by pipeline of work.
Current threads reading	Current number of threads reading source data. It is decremented when blocked by pipeline of work.
Current threads writing	Current number of threads writing data to disk. It is decremented when blocked by pipeline of work.
File bytes written/sec	Rate of bytes written to a multidimensional (MOLAP) file per second.
File rows written/sec	Rate of rows written to a MOLAP file per second.
Memory size bytes	Size of current rows in memory.
Memory size rows	Count of rows in memory.
Rows created/sec	Rate of aggregation rows created. This will be different from rows read/sec if duplicates exist.
Rows merged/sec	Number of rows merged or inserted into an aggregation, per second.
Rows read/sec	Number of rows read from source database per second.
Total rows	Count of rows read from source database.
Total partitions	Accumulating count of partitions processed. This count is incremented if process is successful or not.

See Also

Monitoring Analysis Services Performance

Analysis Server:Proc Aggs Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Proc Aggs** object. The **Proc Aggs** object counters provide information about processing aggregations.

Proc Aggs object counters	**Description**
Current partitions	Current number of partitions being processed.
Memory size bytes	Size of current aggregations in memory. This count is an estimate.
Memory size rows	Size of current aggregations in memory. This count is an estimate.
Rows created/sec	Number of aggregation rows created per second.
Rows merged/sec	Number of rows merged or inserted into an aggregation per second.
Temp file bytes written/sec	Number of bytes written to a temporary file per second. Temporary files are written when aggregations exceed memory limits.
Temp file rows written/sec	Number of temporary file rows written per second. Temporary files are written when aggregations exceed memory limits.
Total partitions	Total number of partitions processed (successfully or otherwise).

See Also

Monitoring Analysis Services Performance

Analysis Server:Proc Indexes Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Proc Indexes** object. The **Proc Indexes** object counters provide information about processing indexes.

Proc Indexes object counters	Description
Current partitions	Current number of partitions being processed.
Rows/sec	Count of rows per second that are working on processing indexes.
Total partitions	Accumulating count of partitions processed. This count is incremented regardless of whether process is successful or not.
Total rows	Accumulating count of rows that are processing indexes.

See Also

Monitoring Analysis Services Performance

Analysis Server:Query Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Query** object. The **Query** object counters provide information about processing on the last query, such as elapsed time in milliseconds of the last query processed.

Query object counters	Description
Avg time/query	Average time per query in milliseconds. This counter is based on queries answered since the last counter measurement.
Bytes sent/sec	Count of bytes sent per second.
Current process thread pool	Current count of threads in processing threads pool. These threads resolve queries.
Current process thread queue length	Current length of queue for work items in the processing thread pool.
Current process threads active	Current number of threads actively working on queries.
Current pyramid operations	Current pyramid operations.
Current queries	Current number of queries being actively worked on.
Current threads	Current number of threads working on queries.
Current worker thread pool	Current count of threads in worker thread pool. These threads answer requests from clients.
Current worker threads active	Current count of worker threads active.
Data bytes/sec	Count of bytes reads from the data file per second.
Data reads/sec	Count of logical I/O reads against the data file per second.

Filter rows excluded/sec	Count of filter rows excluded per second.
Filter rows included/sec	Count of filter rows included per second.
Filtered rows/sec	Count of filtered rows per second.
Index bytes/sec	Count of byte reads from the Index file per second.
Index reads/sec	Count of logical I/O reads against the index file per second.
Map bytes/sec	Count of bytes read from the Map file per second.
Map reads/sec	Count of logical input/output (I/O) reads against the Map file per second.
Network round trips/sec	Rate of network round trips. This includes all client/server communication.
Pyramid operations/sec	Rate of pyramid operations started.
Queries answered/sec	Count of queries being answered per second.
Queries from cache direct/sec	Count of queries from cache direct per second.
Queries from cache filtered/sec	Count of per queries from cache filtered per second.
Queries from file/sec	Rate of queries answered from files.
Queries requested/sec	Count of query requests arriving at the server per second.
Rows read/sec	Count of rows read per second.
Rows sent/sec	Count of rows sent per second.
Total bytes sent	Accumulating count of bytes sent.
Total network round trips	Total network round trips. This includes all client/server communication.
Total pyramid operations	Accumulating count of pyramid operations.
Total queries answered	Accumulating count of queries answered.
Total queries from cache direct	Accumulating count of queries derived directly from cache. Note that this is per

	partition.
Total queries from cache filtered	Accumulating count of queries from cache filtered.
Total queries from file	Accumulating count of queries from files.
Total queries requested	Accumulating count of queries requested.
Total rows sent	Accumulating count of rows sent.

See Also

Monitoring Analysis Services Performance

Analysis Server:Query Dims Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Query Dims** object. The **Query Dims** object counters provide information about processing queries dimensions.

Query Dims object counters	Description
Bytes/sec	Count of sending bytes per second.
Current requests	Current number of requests for part of the member tree or for member properties.
Members/sec	Count of sending members per second.
Requests/sec	Rate of requests for part of the member tree or for member properties. Consider increasing Large Level Threshold if this counter is excessive.
Total bytes	Accumulating count of bytes sent.
Total members	Total number of requests for part of the member tree or for member properties.
Total requests	Accumulating count of requests.
Total VLDM requests	Accumulating count of Very Large Dimension Manager (VLDM) requests sent.
VLDM requests/sec	Count of VLDM requests per second.

See Also

Monitoring Analysis Services Performance

Analysis Server:Startup Object

The following table lists performance counters from the Microsoft® SQL Server™ 2000 Analysis Services **Startup** object. The **Startup** object counters provide information about processing startups.

Startup object counters	Description
Bytes/sec	Count of bytes per second.
Members/sec	Count of loading members per second.
Properties/sec	Count of loading properties per second.
Server uptime	Accumulating seconds since the server was started.
Total bytes	Accumulating count of bytes.
Total dimensions	Accumulating count of dimensions loaded.
Total members	Accumulating count of members loaded.
Total properties	Accumulating count of properties loaded.

See Also

Monitoring Analysis Services Performance

Analysis Services

Automating and Scheduling Administrative Tasks

Microsoft® SQL Server™ 2000 Analysis Services integrates with Data Transformation Services (DTS) to help you administer your OLAP and data mining data. You can use DTS to automate processing, and you can use a batch file to automate archive and restoration of Analysis Services databases. The following topics describe how to automate these administrative tasks by using DTS or the command prompt.

Topic	Description
Processing Objects Using Data Transformation Services	Describes how Analysis Services works with DTS to process objects.
Creating Predictions Using Data Transformation Services	Describes how DTS can be used to create a task that creates output tables containing the predictive column results from a data mining model analysis.

Processing Objects Using Data Transformation Services

Use the Analysis Services Processing task in a Data Transformation Services (DTS) package to perform processing of one or more cubes or other objects defined in Microsoft® SQL Server™ 2000 Analysis Services. Use DTS to extract, transform, and consolidate data from disparate sources into single or multiple destinations. For more information about using DTS with Analysis Services, see Using Data Transformation Services with Analysis Services. For more information about DTS, see DTS Overview.

After you define the package with the task, you can schedule the package to execute at a future date and time. The package can be executed once or repeatedly at a periodic interval.

Note To use the Analysis Services Processing task, you must install SQL Server 7.0 or SQL Server 2000, including SQL Server Enterprise Manager, as well as Analysis Manager on the server computer on which you want to execute the package containing the task.

To process one or more objects using the Analysis Services Processing task, complete the following steps.

1. Create a DTS package that processes objects in Analysis Services.
    a. Create an Analysis Services processing task.

    b. (Optional.) Add connections and other tasks to the package.

    c. Save the package.
2. Schedule and execute the package.

For more information about steps a and b, see Creating an Analysis Services Processing Task and Adding Connections and Other Tasks to the Package.

Creating an Analysis Services Processing Task

To create an Analysis Services Processing task, use the **Analysis Services Processing Task** dialog box, which is accessed through Data Transformation Services (DTS) Designer in SQL Server Enterprise Manager.

To create an Analysis Services Processing task

Adding Connections and Other Tasks to the Package

In a Data Transformation Services (DTS) package, you can add connections and tasks of various types in addition to the Analysis Services Processing task. The specific connections and tasks you may need depend on your unique requirements and circumstances. The following table lists some common examples.

If the Analysis Services Processing task ...	You can add ...	That executes ...	Because ...
Processes a cube with the **Process** option	• A connection to a Microsoft® SQL Server™ 2000 containing an OLTP database, and • A connection to a SQL Server containing a data warehouse, and • A Data	Before the Analysis Services Processing task	You can schedule and periodically execute the package to automatically update the data warehouse and add current information to the cube.

	Driven Query task to update the data warehouse from the OLTP database		
Processes a cube or partition with the **Incremental update** option	A Microsoft ActiveX® Script task that modifies one or more of the properties of the Analysis Services Processing task. These properties include **Datasource**, **FactTable**, and **Filter.**	Before the Analysis Services Processing task	A package that is executed repeatedly to perform periodic incremental updates usually requires that one or more of these properties be changed for each execution. For more information, see Changing Properties of an Analysis Services Processing Task.
Processes any object with any option	A Send Mail task configured so that the task executes if the preceding task fails	After the Analysis Services Processing task	An administrator will be notified automatically if the Analysis Services Processing task fails.

Note In a package, you do not need to add connections to the data sources of databases, cubes, and partitions defined in SQL Server 2000 Analysis Services. The Analysis Services Processing task establishes the required connections.

After you add the connections and tasks, you can organize them into an orderly workflow by adding precedence constraints.

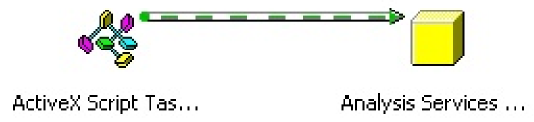
See Also

DTS Overview

Changing Properties of an Analysis Services Processing Task

You can change some properties of an Analysis Services Processing task by including in its package a Microsoft® ActiveX® Script task that executes before the Analysis Services Processing task. The ActiveX Script task can change property values every time the package is executed or it can change them depending on conditional logic. An ActiveX Script task that updates properties for every execution is particularly useful prior to an Analysis Services Processing task that performs incremental updates of a cube or partition. These incremental updates usually require that the filter or fact table be changed for each package execution.

After you add the two tasks to a package and connect them with a success precedence constraint, the Data Transformation Services (DTS) Designer design sheet looks like this:



The following properties of the Analysis Services Processing task can be changed. Note that all property names are case sensitive.

- **ProcessOption**. This is the processing option for the object or folder processed by the task. The following table lists valid values for **ProcessOption** depending upon the object selected.

Object or folder	Processing option	ProcessOption property value	Description
Database	**Process**	0	Completely processes all cubes, partitions, and dimensions in the database.
Cubes	**Process**	0	Completely processes all

folder			cubes in the folder.
	Refresh data	1	For each cube in the folder, performs a refresh data operation if possible; otherwise, completely processes the cube.
Cube with single partition*	**Process**	0	Completely processes the cube, including structural changes. This is the most thorough type of cube processing.
	Refresh data	1	Reloads cube data and recalculates aggregations. This option processes changes to existing source data but not the addition of source data. This option does not process structural cube changes such as new dimensions, levels, or measures.
	Incremental update	2	Adds new data to cube and updates aggregations. This option processes the addition of source data. This option does not process changes to the cube's structure or existing source data.
Cube with multiple partitions*	**Process**	0	Completely processes the cube, including structural changes. This option is more thorough than the **Refresh data** option.

	Refresh data	1	Reloads cube data and recalculates aggregations. This option processes changes to existing source data but not the addition of source data. This option does not process structural cube changes such as new dimensions, levels, or measures.
Partition, including remote partitions*	**Process**	0	Reloads partition data and recalculates aggregations. This option processes changes to existing source data but not the addition of source data. This option does not process structural changes to the parent cube such as new dimensions, levels, or measures.
	Incremental update	2	Adds new data to a partition and updates aggregations. This option processes the addition of source data. This option does not process changes to the structure of the parent cube or existing source data of the partition.
Linked cube	**Process**	0	Completely processes the linked cube.
Virtual	**Process**	0	Completely processes the

cube			virtual cube.
Dimensions folder	**Process**	0	Completely processes all dimensions in the folder.
	Incremental update	2	For each dimension in the folder, performs an incremental update operation if possible; otherwise, completely processes the dimension.
Shared dimension	**Rebuild the dimension structure**	0	Completely processes the dimension, including structural changes. This option is more thorough than the **Incremental update** option.
	Incremental update	2	Processes the addition of members (that is, rows) to the dimension table. This method does not process changes to the structure of the dimension or relationships among members.
Virtual dimension	**Rebuild the dimension structure**	0	Completely processes the virtual dimension.
Mining Models folder	**Process**	0	Completely processes mining models in the folder.
	Refresh data	1	For each mining model in the folder, performs a refresh data operation if possible; otherwise, completely processes the

			mining model.
Mining model**	**Process**	0	Completely processes the mining model.
	Refresh data	1	Adds new data to the source data of the mining model and updates nodes. This option does not process changes to the structure of the mining model or existing source data.

* An additional property, **IncrementallyUpdateDimensions**, is available when this object is selected. Its data type is Boolean.
** The **TrainingQuery** property is an additional string property available when a mining model object is selected.

- **Datasource**. This is the data source used for an incremental update of a cube or partition. Valid values are strings that contain data source names as they appear in the Analysis Manager tree pane.

- **FactTable**. This is the fact table used for an incremental update of a cube or partition. Valid values are strings that contain fact table names.

- **Filter**. This is an expression that limits the fact table records selected for the incremental update of a cube or partition. Valid values are strings that contain valid filters. For more information about filters, see Partition Filters and Incremental Update Filters.

In addition to specifying properties and their values, the ActiveX Script task must name the Analysis Services Processing task to change. Task names are displayed in the **Analysis Services Processing Task** dialog box.

The following example shows how the task name DTSTask_DTSOlapProcess.Task_1 is used to indicate which Analysis Services Processing task is changed.

Example

The following code, written in Microsoft Visual Basic® Scripting Edition, is used in an ActiveX Script task that modifies the properties of an Analysis Services Processing task. The Analysis Services Processing task incrementally updates the sample Sales cube included in Microsoft SQL Server™ 2000 Analysis Services. The incremental update is executed monthly throughout 1998 except in December.

The code selects a fact table different than the original fact table for the Sales cube (**sales_fact_1997**). It also creates a different filter each month.

Note The code specifies the **sales_fact_1998** fact table in the sample **FoodMart 2000** database. This table includes data for only 1998 (except December). Before executing the package containing this code, set your computer's clock to a date between January 1, 1998 and November 30, 1998. Immediately after the package executes, reset the clock to the current date.

```
Function Main()
  Dim pkg
  Dim task
  Dim props
  Dim currentyear
  Dim currentmonth
  Set pkg = DTSGlobalVariables.parent
  Set task = pkg.Tasks("DTSTask_DTSOlapProcess.Task_1")
  Set props = task.Properties
  props("ProcessOption").Value = 2
  props("Datasource").Value = "FoodMart"
  props("FactTable").Value = "sales_fact_1998"
  ' Create filter based on current year and month.
  currentyear = YEAR(NOW)
  currentmonth = MONTH(NOW)
  props("Filter").Value = "([sales_fact_1998].[time_id] " _
     & "IN (SELECT [time_id] FROM [time_by_day] " _
     & "WHERE [time_by_day].[the_year] = " _
     & currentyear  _
     & " AND [time_by_day].[month_of_year] = " _
```

```
        & currentmonth & "))"
   Main = DTSTaskExecResult_Success
End Function
```

After the incremental update is complete, you must merge the selected rows from **sales_fact_1998** into **sales_fact_1997**. For more information, see Fact Table Considerations When Merging Partitions.

See Also

ActiveX Script Task

Creating an Analysis Services Processing Task

Processing Cubes

Creating Predictions Using Data Transformation Services

Use the **Data Mining Prediction Query Task** dialog box in a Data Transformation Services (DTS) package to run prediction queries based on a data mining model defined in Microsoft® SQL Server™ 2000 Analysis Services. Such queries can be used to create output tables that store the predictive results from a mining model analysis.

After you define the DTS package with the task, you can schedule the package to execute at a future date and time. The package can be executed once or at periodic intervals.

Note To use the Data Mining Prediction Query task, you must install SQL Server 7.0 or SQL Server 2000, including SQL Server Enterprise Manager, as well as Analysis Manager on the server computer on which you want to execute the package containing the task.

To create a Relational Data Mining Prediction Query task

Analysis Services

Administrator's Reference

The Administrator's Reference contains technical information about Microsoft® SQL Server™ 2000 Analysis Services, such as specifications and limits. The Analysis Manager Help topics accessed by the F1 key and **Help** buttons are also inluded.

Analysis Services

Specifications and Limits

This topic contains information about supported relational database management systems, object specifications, and source column data types in OLE DB.

Supported Relational Database Products

Microsoft® SQL Server™ 2000 Analysis Services supports the following relational database products as data sources:

- SQL Server versions 6.5 and later
- Microsoft Access 97 and later
- Oracle versions 7.3 and 8.0

  Note Text columns in Oracle tables should be variable length column type when using the OLE DB Provider for Oracle. Use the OLE DB for Provider for ODBC if Oracle tables contain text columns of fixed length column type.

Specifications

The following specifications apply to Analysis Services.

Item	Specification
Dimensions in a database	65,535 maximum, regardless of the number of cubes or whether dimensions are shared or private
Levels in a database	65,535 maximum
Cubes in a virtual cube	64 maximum
Measures in a cube	1,024 maximum
Measures in a virtual cube	2,048 maximum
Dimensions in a cube	128 maximum
Levels in a cube	256 maximum

Levels in a dimension	64 maximum
Members in a dimension	2^31-1 = 2,147,483,647 maximum
Members in a parent	64,000 maximum
Calculated members (server defined) in a cube	65,535 maximum
Calculated members in a parent measure in session context	31,743 maximum
Calculated members in a parent measure in query context	31,743 maximum
Calculated members in a parent dimension member in session context	759 maximum
Calculated members in a parent dimension member in query context	759 maximum
Aggregations per partition	65,535 maximum
Cells returned by a query	2^31-1 = 2,147,483,647 cells maximum Although cubes can be larger than this limit, a query that requests more than 2^31-1 cells from a cube will fail.
Record size for source database table	64 kilobytes (KB) maximum
Length of object name (except dimension name)	50 characters maximum when using Analysis Manager 24 characters maximum when using PivotTable® Service
Length of dimension name	24 characters maximum
Length of aggregation prefix	50 characters maximum
Maximum number of distinct states in a data mining model attribute column	255, after which the column becomes MODEL_EXISTENCE_ONLY For more information, see Data Mining Columns

Note The term character in this documentation refers to a UNICODE character.

Source Column Data Types

The data type of the source column for a measure must be numeric except when the **Count** or **Distinct Count** aggregate function is used.

The data type of the source column for a dimension level must be string or numeric (except currency).

OLE DB Data Types

Columns of the following OLE DB data types may be used as measures or dimension levels. Columns types that are marked with an asterisk are supported data types for data mining model columns.

- DBTYPE_BOOL*
- DBTYPE_I1
- DBTYPE_I2
- DBTYPE_I4*
- DBTYPE_UI1
- DBTYPE_UI2
- DBTYPE_UI4
- DBTYPE_I8

- DBTYPE_UI8
- DBTYPE_R4
- DBTYPE_R8* (Note: Data mining model column inputs of this type will be cast to a 4 byte floating point number)
- DBTYPE_DECIMAL
- DBTYPE_NUMERIC
- DBTYPE_VARNUMERIC
- DBTYPE_CY
- DBTYPE_DATE*
- DBTYPE_DBDATE
- DBTYPE_DBTIME
- DBTYPE_DBTIMESTAMP

Columns of the following OLE DB data types may be used as measures only if the **Count** or **Distinct Count** aggregate function is used. These data types may be used as dimension levels.

- DBTYPE_BYTES
- DBTYPE_HCHAPTER*

- DBTYPE_STR
- DBTYPE_WSTR*

External Limitations

Limitations imposed by other technologies, such as the RDBMS being used, may limit some features of Analysis Services. For example, when merging two partitions containing a large number (> 100) of aggregations, you may receive an error message indicating that the maximum number of ODBC Access 97 File Sharing lock counts has been exceeded. This number is controlled by the Access 97 **MaxLocksPerFile** registry entry, not by any configuration parameter in Analysis Services.

Other such external limitations may apply as well.

Analysis Services

SQL

Microsoft® SQL Server™ 2000 Analysis Services supports SQL queries or clauses in filters. A filter is the section of a Structured Query Language (SQL) SELECT statement that follows the WHERE keyword. That is, it is the list of predicates that make up the WHERE clause of the SELECT statement. The following topics describe how filters are used in Analysis Services.

Topic	Description
Partition Filters and Incremental Update Filters	Describes the filters specify the data to be incorporated into the partitions of a cube.
Dimension Filters	Describes the filters that specify which members from a dimension table will be used to build a dimension.
Drillthrough Filters	Describes the filter that specifies which aggregation elements are returned when users drill through to the source data. For more information about using drillthrough, see Using DRILLTHROUGH to Retrieve Source Data.

You can use a filter to limit the following:

- The dataset of a partition
- The dataset of a dimension
- The rows used during an incremental update
- The rows returned by drillthrough

In PivotTable® Service, SQL can be used to query data, build local cubes, and build data mining models. For more information, see PivotTable Service.

Analysis Services is both a multidimensional data provider and a tabular data provider. Therefore, executing a query returns either a multidimensional dataset or a flattened rowset, depending on the query language dialect used. Analysis Services recognizes two dialects: SQL and Multidimensional Expressions (MDX). For more information, see SQL in Analysis Services.

See Also

PivotTable Service

Partition Filters and Incremental Update Filters

Dimension Filters

Drillthrough Filters

Partition Filters and Incremental Update Filters

You can use filters to specify the data that you want to incorporate into the partitions of a cube. You can specify a filter for your partition data or incremental update by entering the criteria expression of an SQL WHERE clause.

Note When entering a filter clause, the WHERE keyword is not required.

The filter expression you enter is used in a pass-through statement to be executed by the source database. The filter is not verified for syntax until the partition or incremental update is processed. Regardless of the complexity of your filter, data is retrieved only from the fact table for use in the partition. The default value for a partition filter is derived from the **Source Table Filter** property for the parent cube. This property is accessed in the properties pane of Cube Editor.

The simplest filters are based on one or more columns in the fact table. For example, to select rows for only the West region from the **Location** fact table, use the following filter:

```
"location"."region"='West'
```

Note Because the 1992 ISO and ANSI standards for SQL specify that double quotation marks (") should be used as delimiters for table and column names, the preceding example uses this convention. Microsoft® SQL Server™ 2000 and Microsoft Access standards also support brackets ([ ]).

More complicated filters are also possible. Because dimension tables are inner-joined to the fact table, a filter can include criteria applied against the fact table or any dimension table used by dimensions in the partition. However, to filter on dimension tables, the column names from the dimension tables must be contained in a nested SELECT statement, and the underlying database must support nested SELECT statements. If the underlying database supports nested SELECT statements, you can use tables that are not referenced by the cube definition in your filter, but you must specify the join to the fact table in your filter.

Important Whenever necessary to avoid ambiguity, use a qualified expression. For example, if a column name appears in multiple tables, include the table name in the expression.

The following examples of filters can be used in a partition for a cube that is based on the sample **FoodMart 2000** database, where the cube includes the **sales_fact_1997** fact table and the **time_by_day** dimension table (and possibly other dimension tables):

- This filter uses the fact table (518 is the **time_id** for June 1, 1997, and 547 is the **time_id** for June 30, 1997):
  ```
  "sales_fact_1997"."time_id" BETWEEN 518 AND 547
  ```

- The following equivalent filter uses the fact table and the **time_by_day** dimension table in a nested query:
  ```
  "sales_fact_1997"."time_id" IN
  (SELECT "time_id" FROM "time_by_day"
  WHERE "time_by_day"."the_year" = 1997
  AND "time_by_day"."the_month" = 'June')
  ```

You can set filters in either the **Advanced Settings** dialog box or in the Incremental Update Wizard. The **Advanced Settings** dialog box appears when you click **Advanced** in the last step of the Partition Wizard or in the **Convert to Partition** dialog box.

Using Filters with Overlapping Partition Data

Incorrect results can be returned from cubes whose partitions contain overlapping data as a result of filter statements that are not mutually exclusive. For more information, see Managing Partitions.

You must ensure that no data is duplicated among multiple partitions, and that no data is duplicated within a partition. For example, these sets of filters are mutually exclusive within each set.

Examples

A. Making the Years 1997 and 1998 Exclusive of Each Other

"SaleYear" = 1997
"SaleYear" = 1998

B. Making Continents in the Geography Dimension Exclusive of E

"Continent" = 'NorthAmerica'
"Continent" = 'Europe'
"Continent" = 'SouthAmerica'

C. Creating Expressions That Are Exclusive of Each Other

"Country" = 'USA'
"Country" = 'Mexico'
("Country" <> 'USA' AND "Country" <> 'Mexico')

D. Creating Exclusive Filters That Include All Partition Data

When you create mutually exclusive filters for partitions, ensure that the combined partition data includes all data you want for the cube. The following filter expressions, the first in one of two partitions in a cube and the second in the other, yield incorrect query results from the cube data because the USA data is included in both partitions.

"Continent" = 'NorthAmerica'
"Country" = 'USA'

See Also

Partitions

Dimension Filters

You can use a dimension filter to specify which members of the source table are used to build that dimension. This topic provides examples of how to do this in different scenarios.

The following example demonstrates how to limit the members of the Geography dimension to those members in the USA.

Examples

A. Limiting Geography Dimension Members to Members of the USA Country Level

"Customer.Country" = 'USA'

B. Using Boolean Operators to Combine Selected Sets

"Product.ProductFamily" = 'Drink' and "Store Type.Store Type" = 'Supermarket'

"Store.Store City" = 'Seattle' or "Store.Store City" = 'Tacoma'

See Also

SourceTableFilter (Dimension Interface)

Drillthrough Filters

You can use a drillthrough filter to limit the number of rows returned when users use the **Drillthrough** qualifier in a Multidimensional Expressions (MDX) query. Because the component elements for a member (that is, the elements that make up the aggregation of the member) from the fact table can be numerous, a mechanism is needed to limit the number of rows that come back as a result of a drillthrough.

Examples

Using a Filter with the Drillthrough Function

The following example limits the results returned by the **Drillthrough** function to those customers who live in the city of Olympia:

```
"Customers.City" = "Olympia"
```

See Also

DrillThroughFilter

Drillthrough Options Dialog Box

Filter Tab (Drillthrough Options Dialog Box)

Analysis Services

User Interface Help Reference

This section includes Help topics for Microsoft® SQL Server™ 2000 Analysis Services. These topics are also available from the user interface by pressing the F1 key or by clicking **Help** in wizard steps and dialog boxes.

Section	**Contains**
Analysis Services Icons	Icons used in Analysis Services
Wizards	Help topics for wizards
Dialog Boxes	Help topics for dialog boxes

Analysis Services

Analysis Services Icons

The following table lists links to topics that identify icons used in Analysis Manager, editors, and browsers in Microsoft® SQL Server™ 2000 Analysis Services.

Topic	Description
Analysis Manager Icons	Icons used in Analysis Manager
Cube Editor Icons	Icons used in Cube Editor
Dimension Browser Icons	Icons used in Dimension Browser
Dimension Editor Icons	Icons used in Dimension Editor
OLAP Mining Model Editor Icons	Icons used in OLAP Mining Model Editor
Relational Mining Model Editor Icons	Icons used in Relational Mining Model Editor
Virtual Cube Editor Icons	Icons used in Virtual Cube Editor

Analysis Manager Icons

The following table lists the icons displayed in the tree pane of Analysis Manager.

Icon	Description
	Active server
	Inactive server
	Unknown server
	Database
	Data source
	Regular cube
	Virtual cube
	Linked cube
	Local partition
	Remote partition
	Regular (shared) dimension
	Virtual dimension
	Parent-child dimension
	Mining model dimension
	Relational mining model
	OLAP mining model
	Cube role
	Database role
	Mining model role

Cube Editor Icons

The following table lists the icons used in the tree pane of Cube Editor. The See Also links provide information about toolbar icons used in Cube Editor.

Icon	Description
	Regular (shared) dimension
	Virtual dimension
	Parent-child dimension
	Mining model dimension
	Private dimension
	(All) Level
	Level 1
	Level 2
	Level 3
	Level 4
	Level 5
	Level 6
	Level 7
	Level 8
	Level 9
	Level 10
	Level 11
	Level 12
	Level 13
	Level 14
	Level 15
	Level 16
	Member property
	Measure
	Calculated member

	Calculated cells
	Action
{..}	Named set

See Also

Toolbar (Cube Editor Data View)

Toolbar (Cube Editor Schema View)

Dimension Browser Icons

The following table lists the icons used in the tree pane of Dimension Browser. In addition to the following icons, the unary operator precedes the icon for each member shown in the tree pane.

Icon	Description
●	Member
	Custom member
	Calculated member
	Measure
	Calculated measure

Dimension Editor Icons

The following table lists the icons used in the tree pane of Dimension Editor. The See Also links provide information about toolbar icons used in Dimension Editor.

Icon	Description
	Regular (shared) dimension
	Virtual dimension
	Parent-child dimension
	Mining model dimension
	Level 1
	Level 2
	Level 3
	Level 4
	Level 5
	Level 6
	Level 7
	Level 8
	Level 9
	Level 10
	Level 11
	Level 12
	Level 13
	Level 14
	Level 15
	Level 16
	Member property

See Also

Toolbar (Dimension Editor Data View)

Toolbar (Dimension Editor Schema View)

OLAP Mining Model Editor Icons

The following table lists the icons used in the tree pane of OLAP Mining Model Editor. The See Also links provide information about toolbar icons used in OLAP Mining Model Editor.

Icon	Description
	OLAP mining model
	Mining model dimension
	Case dimension
	Predictable case dimension
	Predictable dimension
	Case level
	Predictable case level
	Predictable level
	Case measure
	Predictable case measure
	Predictable measure
	Case member property
	Predictable case member property
	Predictable member property

See Also

Toolbar (OLAP Mining Model Editor)

Relational Mining Model Editor Icons

The following table lists the icons used in the tree pane of Relational Mining Model Editor. The See Also links provide information about toolbar icons used in Relational Mining Model Editor.

Icon	Description
	Relational mining model
	Key column
	Input column
	Predictable column
	Input and predictable column

See Also

Toolbar (Relational Mining Model Editor)

Virtual Cube Editor Icons

The following table lists the icons used in the tree pane of Virtual Cube Editor. The See Also links provide information about toolbar icons used in Virtual Cube Editor.

Icon	Description
	Regular (shared) dimension
	Virtual dimension
	Parent-child dimension
	Mining model dimension
	Private dimension
	(All) Level
	Level 1
	Level 2
	Level 3
	Level 4
	Level 5
	Level 6
	Level 7
	Level 8
	Level 9
	Level 10
	Level 11
	Level 12
	Level 13
	Level 14
	Level 15
	Level 16
	Member property
	Measure
	Calculated member

	Calculated cells
	Action
{..}	Named set

See Also

Toolbar (Virtual Cube Editor)

Analysis Services

Wizards

This section contains Help topics that are available from the user interface by pressing F1 or by clicking **Help** in wizards.

See Also

Dialog Boxes

Action Wizard

Use this wizard to create actions for a cube. Actions allow client applications to trigger various operations in response to specific browsing activities.

The Action Wizard appears when you perform any of the following actions:

- In Cube Editor, on the **Insert** menu, click **Action**.
- In the Cube Editor tree pane, right-click the Actions folder, and then click **New Action**.
- In Virtual Cube Editor, on the **Insert** menu, click **Action**.
- In the Virtual Cube Editor tree pane, right-click the Actions folder, and then click **New Action**.
- Right-click an existing action in Cube Editor or Virtual Cube Editor, and then click **Edit**.

The Action Wizard has the following steps:

- Select a target object for the action.
- Select the type of action to create.
- Define the syntax for the action.
- Name and save your action.

See Also

Actions

Introduction (Action Wizard)

Use this wizard to create an action.

Options

Skip this screen in the future

Select to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

See Also

Actions

Select Target (Action Wizard)

In this step of the wizard, you select the object that launches the action. Depending on the object you define as the target, you may be prompted to specify additional options within this step.

Options

Target

Select the object that you want to be the target of the action. The target launches the action. If you choose **This cube**, **Cells in this cube**, or **Sets in this cube**, you can move to the next step of the wizard. If you choose **A dimension in this cube** or **A level in this cube**, you can change the target of the action.

Dimension

Select the dimension in which you want to define the action, or specify a dimension level within the cube as the target object.

Levels

Select a level in the chosen dimension.

Define the target as

- **Members of the selected dimension**

  Select to make the action available if end users select any member of a dimension while browsing.

- **The dimension object**

  Select to make the action available only if you select the heading that represents the dimension while you are browsing.

- **Members of the selected level**

  Select to make the action available for launching if you select any member of a level while you are browsing

- **The level object**

  Select to make the action available for launching only if users select the heading that represents the level while browsing.

See Also

Actions

Named Sets

Select the Action Type (Action Wizard)

In this step of the wizard, you select the type of action to be launched when an end user selects the target object in the client application while browsing the cube.

Options

Type

Select from the following action types.

Select	To
Command Line	Create an action that executes an MS-DOS command prompt. For example, you can use the path C:\Winnt\Notepad.exe to start the Notepad application in Windows NT® 4.0 or Windows® 2000.
Statement	Create an action that executes an SQL statement through OLE DB from the client application.
HTML	Create an action that executes an HTML script within the default Web browser.
URL	Create an action that navigates to a Web address and displays its respective page in the default Web browser.
Data set	Create an action that returns a multidimensional data set.
Rowset	Create an action that returns a set of tabular rows.
Proprietary	Create an action that enables the client application to perform a custom action.

Sample

View an example of the syntax for the type of action you select.

See Also

Actions

Define the Action Syntax (Action Wizard)

In this step of the wizard, you specify the syntax for the action. The syntax is in the form of a Multidimensional Expressions (MDX) expression and can include literal expressions enclosed by quotation marks. For example, if you want to create a URL action that qualifies a specific address, your syntax can use a quoted expression for the address followed by a plus sign (+) and an MDX function:

```
"http://MyIntranetServer/Sales.asp?"+CustomerDimension.CurrentMe
```

Options

Syntax

Type the syntax directly into the **Syntax** box, or click **MDX Builder** to display MDX Builder, which can help you define the syntax.

Sample

View an example of the syntax for the type of action you selected in the **Select the Action Type** step of the wizard.

See Also

Actions

MDX Builder

Finish (Action Wizard)

In this final step of the wizard, you supply a name for the action. After naming your action and finishing the wizard, you can save and test your action in Cube Editor or Virtual Cube Editor.

Note This wizard saves the action, but you must also save the cube or virtual cube with which the action is associated. You cannot select or test actions in the data viewing pane until after you save them in the editor.

How to test an action

Calculated Cells Wizard

Use this wizard to create calculated cells. Calculated cells enable you to use a Multidimensional Expressions (MDX) expression to override values for cells in a specific area of a cube.

The Calculated Cells Wizard appears when you perform any of the following actions:

- In the Cube Editor or Virtual Cube Editor tree pane, right-click the Calculated Cells folder, and then click **New Calculated Cells**.
- In the Cube Editor or Virtual Cube Editor, on the **Insert** menu, click **Calculated Cells**.
- Right-click an existing calculated cells definition in the Cube Editor or Virtual Cube Editor, and then click **New Calculated Cells** or **Edit**.

The Calculated Cells Wizard has the following steps:

- Define the calculation subcube.
- Define the calculation condition.
- Define the calculation formula.
- Name and save the calculated cells definition.

See Also

Calculated Cells

Introduction (Calculated Cells Wizard)

Use this wizard to create calculated cells. The wizard helps you define the calculation subcube, condition, and formula for calculated cells.

Options

Skip this screen in the future

Select this option to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

Define the Calculation Subcube (Calculated Cells Wizard)

In this step of the wizard, you define the calculation subcube. The subcube defines the scope of the calculation formula. To restrict the subcube, select a dimension from the **Dimensions** box and a member selection operator from the **Members Set** box.

Options

Dimensions

Select each dimension for which you want to specify a Members Set.

Members Set

After you select a dimension, select the members set to which the formula applies. You can press DELETE while a dimension is selected to reset it to All members. The available options depend on the type of dimension selected.

The following table lists options that apply to the Measures dimension.

Select	To
All members	Select all members in a dimension. This choice results in an empty set that is interpreted as all members, excluding calculated members, of the Measures dimension. This is the default for the Measures dimension.
A single measure	Select a single measure from the list displayed.
Custom MDX expression	Build a Multidimensional Expressions (MDX) expression either by typing it under **MDX Expression** or by clicking **MDX**

	Builder to use MDX Builder.

The following table contains options that apply to all other dimensions.

Select	To
All members	Select all members in a dimension. This choice results in an empty set that is interpreted as all members, excluding calculated members, of the dimension. This is the default for each dimension.
A single member	Select a single member from the list displayed.
A single level	Select a single level in the dimension from the list displayed.
Descendants of a member	Display a tree view of the dimension. Expand the tree and select a single member.
Descendants of a member at a level	Select a level for a member. Expand the tree under **Member** to select a member, and then under **Level** select a single level.
Custom MDX expression	Build a MDX expression either by typing it under **MDX Expression** or by clicking **MDX Builder** to use MDX Builder.

See Also

Calculated Cells

Members, Tuples, and Sets

Define the Calculation Condition (Calculated Cells Wizard)

In this optional step of the wizard, you can specify a logical Multidimensional Expressions (MDX) expression that limits the scope for a calculated cells formula. The calculated cells formula is applied only to cells of the calculation subcube (defined in the previous step) that meet the condition.

Options

Apply the calculation formula to the entire calculation subcube

Choose this option if you are not adding any calculation condition. The calculated cells formula then applies to all cells in the subcube.

Apply the calculation formula to cells in the calculation subcube that meet the following condition

Choose this option to add a Boolean condition. Choosing this option disables the **Next** button until after you add an MDX expression. You can either type an expression or use MDX Builder to construct it.

MDX Expression

Type an MDX expression that evaluates to a single Boolean value. The calculated cells formula applies only to those cells for which the expression evaluates to TRUE. The actual cell value is used for any cell for which the expression evaluates to FALSE.

MDX Builder

Click this to use MDX Builder to construct the expression.

Define the Calculation Formula (Calculated Cells Wizard)

In this step of the wizard, you construct the calculation formula that provides the value for each cell in the calculation subcube. The **Next** button is enabled only after you enter a calculation formula. When you click **Next**, the wizard performs a syntax check and proceeds to the next step only if there are no syntax errors. The wizard displays a warning if the syntax check fails.

Options

MDX Expression

Type a Multidimensional Expressions (MDX) expression that evaluates to a single Boolean value.

MDX Builder

Click this to use MDX Builder to construct the expression.

Finish (Calculated Cells Wizard)

In this step of the wizard, you name and save the calculated cells definition.

Options

Name

Type a unique name. You can enter a maximum of 50 characters for the calculated cells definition name. A calculated cells definition name must begin with a letter of the alphabet. The name cannot be the same as the name of an existing calculated member, calculated cells definition, named set, or action.

Summary

Review the calculation subcube.

Finish

Click to save your calculated cells definition and exit the wizard.

Cube Wizard

Use this wizard to build a multidimensional cube from relational data. After you build and process the cube, you can browse its data in Cube Browser. If you do not process the cube, you will see sample data in Cube Browser.

The Cube Wizard appears when you right-click the Cubes folder in the Analysis Manager tree pane or right-click an existing cube, point to **New Cube**, and then click **Wizard**.

The Cube Wizard has the following steps:

- Select the data source and fact table for your cube.
- Define the measures for your cube.
- Select the dimensions for your cube.
- Name and save your cube.
- Use Cube Editor to further define your cube.

Introduction (Cube Wizard)

Use this wizard to create a cube. The wizard helps you select the data source, fact table, measures, and dimensions for a new cube.

Options

Skip this screen in the future

Select this option to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

See Also

Building and Processing Cubes

Cubes

Select a Fact Table (Cube Wizard)

In this step of the wizard, you select a fact table. A fact table contains the numeric data that cube users want to analyze.

The wizard displays a list of data sources that are currently available. Click the expand (**+**) button to expand the data source and view the tables contained within.

Options

Data sources and tables

Select a primary fact table for the cube from one of the tables in a data source listed in this box.

Details

View the columns in the table selected in the **Data sources and tables** box.

New Data Source

Choose this option to define a new data source. For more information about selecting a data source, click **Help** in the **Data Link Properties** dialog box.

Browse Data

Choose this option to preview the contents of the table you have selected.

Define Measures (Cube Wizard)

In this step of the wizard, you choose the numeric columns that you want to use as measures for the cube.

Options

Fact table numeric columns

Select a primary fact table for the cube from one of the tables in a data source listed in this box.

Cube measures

View the columns in the table selected in the **Data sources and tables** box.

See Also

Measures

Select Dimensions (Cube Wizard)

In this step of the wizard, you choose dimensions for the cube. You can choose from previously defined dimensions, which are shared (that is, available to other cubes), or start the Dimension Wizard to create a new dimension. If you create a new dimension, you can define it as either shared or private (available only to the cube you are creating).

Options

Shared dimensions

Select previously defined dimensions from the list.

Cube dimensions

View the dimensions you are adding to your cube.

New Dimension

Choose this option to start the Dimension Wizard, which helps you create a new dimension.

See Also

Dimensions

Finish (Cube Wizard)

In this step of the wizard, you name and save the cube. You can also review the structure of the cube.

Options

Cube name

> Type a unique name. You can enter a maximum of 50 characters for the cube name. A cube name must begin with a letter of the alphabet.

Cube structure

> Review the measures and dimensions of the cube under **Cube structure**.

Browse Sample Data

> Choose this option to open Cube Browser and view sample data in the dimensions and measures of the cube. You cannot browse your actual data until you have processed the cube.

Note This option is not available if you have been directed to create joins manually in Cube Editor.

Finish

> Click to save your cube, exit the wizard, and display Cube Editor.

See Also

Building and Processing Cubes

Cubes

Dimension Wizard

Use this wizard to create dimensions for your cubes.

The Dimension Wizard appears when you perform any of the following actions:

- In the Cube Wizard, in the **Select dimensions** step, click **New Dimension**.
- In Cube Editor, on the **Insert** menu, point to **Dimension**, and then click **New**.
- In the Analysis Manager tree pane, right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Wizard**. If you start the Dimension Wizard in this way, you cannot create a private dimension.

The Dimension Wizard has the following steps:

- Choose how to create the dimension. You can create it from:
    - Columns in a single table. This option creates a regular dimension.
    - Columns in multiple tables. This option creates a regular dimension.
    - Two hierarchically related columns. This option creates a parent-child dimension.
    - Member properties. This option creates a virtual dimension.
    - The predictable column of an OLAP mining model. This option creates a data mining dimension.

- Select the table or tables for your dimension. This step is displayed only if you are creating a regular or parent-child dimension.
- Select the dimension type: standard or time. This step is displayed only if you are creating a regular dimension and the dimension table contains a **datetime** column.
- Create time dimension levels. This step is displayed only if you are creating a regular time dimension.
- Edit joins. This step is displayed only if you are creating a regular dimension from multiple tables.
- Select the levels for your dimension. This step is displayed only if you are creating a regular, non-time dimension.
- Specify the member key columns. This step is displayed only if you are creating a regular, non-time dimension.
- Select the columns for a parent-child dimension. This step is displayed only if you are creating a parent-child dimension.
- Select the dimension with the member properties. This step is displayed only if you are creating a virtual dimension.
- Select the levels for the virtual dimension. This step is displayed only if you are creating a virtual dimension.
- Select advanced options for your dimension. This step does not appear if you are creating a data mining dimension.

- Confirm if you want to create a changing dimension. This step is displayed only if you selected this as an advanced option.
- Enable and define custom rollup expressions. This step is displayed only if you selected this as an advanced option.
- Specify how members are ordered and uniquely named. This step is displayed only if you selected this as an advanced option.
- Specify the storage mode and automatically create member groups. This step is displayed only if you selected this as an advanced option.
- Enable writeback capability. This step is displayed only if you are creating a parent-child dimension and you selected this as an advanced option.
- Select an OLAP mining model and predictable column. This step is displayed only if you are creating a dimension from an OLAP mining model.
- Finish. Name your dimension and preview its data.

See Also

Dimensions

Introduction (Dimension Wizard)

The Dimension Wizard takes you through the creation of a dimension for your cube.

Option

Skip this screen in the future

Select to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

Choose How To Create Dimension (Dimension Wizard)

In this step of the wizard, you specify the type of source for the dimension's data.

Options

Star Schema: A single dimension table

Select to create a regular dimension based on a single dimension table. The depth of the dimension depends on the number of levels you select in a later step. Each level is derived from a column. Alternatively, a single **datetime** column can be parsed into multiple levels to create a time dimension.

When the dimension is added to a cube, the dimension table joins to the fact table. If each of the cube's dimensions is based on a single table, the cube has a star schema.

Snowflake Schema: Multiple, related dimension tables

Select to create a regular dimension based on multiple, joined dimension tables. The depth of the dimension depends on the number of levels you select in a later step. Each level is derived from a column.

When the dimension is added to a cube, only one of its dimension tables joins to the fact table. Therefore, the cube has a snowflake schema.

Parent-Child: Two related columns in a single dimension table

Select to create a parent-child dimension, which is based on two columns of the same data type. One column identifies each dimension member, and the other identifies each member's parent. For example, in an **Employee** table, an **Employee Number** column contains an identifier for each employee, and a **Manager Employee Number** column contains the employee number of each employee's manager. A parent-child dimension based on these two columns mirrors an organization chart.

When the dimension is added to a cube, the dimension table joins to the fact table.

Virtual Dimension: The member properties of another dimension

Select to create a virtual dimension that is based on one or more member properties in another dimension.

Mining Model: A predictable column of an OLAP mining model

Select to create a data mining dimension from a predictable column of an OLAP mining model.

See Also

Parent-Child Dimensions

Virtual Dimensions

Select Dimension Table (Dimension Wizard)

In this step of the wizard, you select the table or tables from which you want to create your dimension. If you select multiple tables, all the tables must come from the same data source.

This wizard step is not displayed if you selected **Virtual Dimension: The member properties of another dimension or Mining Model: Predictable column of OLAP mining model** in the second step of the wizard.

Under **Available tables**, the wizard displays a list of the tables in your data source that are available for creating dimensions. Click the expand (+) button to expand the data source and view the tables contained within.

Options

Available tables

Select the table you want to add. If you do not see the table you want to use for the dimension, choose another data source.

Note All the tables you select must come from the same data source. If you add a table from a data source that is different from that of the other selected tables, the list of tables under **Selected tables** will be cleared.

Details

View the columns of the table selected in **Available tables**.

New Data Source

Click to choose a data source that is not listed under **Available tables**.

Browse Data

Click to view data in the table selected in **Available tables**.

Select the Dimension Type (Dimension Wizard)

This step of the wizard appears only if, in the **Select dimension table** step, you selected a table that contains both date-formatted columns and columns of other formats. In this step, you specify the type of dimension you want to create.

Options

Standard dimension

Creates a standard categorical dimension, rather than a time dimension.

Time dimension

Creates a time dimension based on a date-formatted column.

- **Date column**

  If you click **Time dimension**, you must select the date-formatted column from the **Date column** list.

Create Time Dimension Levels (Dimension Wizard)

This step appears only if you are creating a time dimension. In this step, you select the levels for your time dimension. You can also specify a start date for your year.

Options

Select time levels

Click the date levels you want in your time dimension. These levels are created from the date-formatted column you selected in the previous step of the wizard.

Dimension structure

Preview the level structure of the dimension you are creating.

Year starts on

Set the start date for your year.

- **Day**

  Select the day of the month on which the dimension's year starts.

- **Month**

  Select the first month in your dimension's year.

Create and Edit Joins (Dimension Wizard)

This step appears only if you selected **Snowflake Schema: Multiple, related dimension tables** in the second step of the wizard, indicating that you would use more than one table for your dimension.

In this step of the wizard, you can create, edit, and remove joins between your dimension tables. The wizard displays the tables you have selected for your dimension. If two or more tables share a column, the wizard shows that it has automatically created a join by showing a line connecting the matching columns.

Note The Dimension Wizard is designed for use with simple schemas. If you are designing a complex snowflake schema, it is recommended that you use Dimension Editor.

If you need to edit joins but do not have a mouse or pointing device, cancel the Dimension Wizard and create the dimension using Dimension Editor, which provides the **Join** command on the **Insert** menu and the **Remove Join** command on the **Edit** menu.

Select Levels (Dimension Wizard)

In this step of the wizard, you can define certain columns as levels in your dimension. The columns you can use appear under **Available columns**.

This wizard step appears only if you selected **Star Schema: A single dimension table**, **Snowflake Schema: Multiple, related dimension tables**, or **Mining Model: A predictable column of an OLAP mining model** in the second step of the wizard.

Note Your levels must appear in hierarchical order beginning with the most general level of detail. For example, a Geography dimension would be ordered World (All), Continents, Countries, Cities.

Options

Available columns

Click the columns you want to use in your dimension. Use the arrow keys to move selected columns to the dimension levels pane.

Dimension levels

View the levels that will be generated based on the columns selected in **Available columns**. Use the arrow keys to remove unwanted levels or to change their order.

Count level members automatically

Counts the members of the selected level in **Available dimensions**. This is required for cube processing. It may take some time to process levels with a large number of members. To speed up the dimension creation process, clear the **Count level members automatically** check box.

Note If automatic counting of level members is turned off, you must manually enter an estimated number for levels using the Dimension Editor properties pane or respond to individual prompts during cube processing.

See Also

Dimension Structure

Levels and Members

Specify Member Key Columns (Dimension Wizard)

In this step of the wizard, you can change the member key columns for your level members as needed. This is necessary if the default member key columns do not uniquely identify the level members.

This wizard step is only visible if you selected **Star Schema: A single dimension table** or **Snowflake Schema: Multiple, related dimension tables** in the second step of the wizard.

Options

Name

View the name of a defined level.

Member Key Column

(Optional.) Select new key column names for levels to override the default values supplied by the wizard.

See Also

Member Names and Member Keys

Select Columns for Parent-Child Dimension (Dimension Wizard)

In this step of the wizard, you select the two columns upon which the parent-child dimension is based. You can also select a column to provide member names.

This wizard step is displayed only if, in the second step of the wizard, you selected **Parent-Child: Two related columns in a single dimension table**.

Options

Member key

Select the column that uniquely identifies each member. The column must have the same data type as the column you select in the **Parent key** box.

Parent key

Select the column that uniquely identifies the parent of each member. This column defines the lineage relationships among the members. For each member, it determines the parent member. The column must have the same data type as the column you select in the **Member key** box.

Member name

Select the column that provides member names to be displayed to end users as they browse cubes.

See Also

Parent-Child Dimensions

Select Dimension with Member Properties (Dimension Wizard)

In this step of the wizard, you select the dimension on which the virtual dimension will be based. This dimension must contain all the member properties on which the virtual dimension's levels will be based.

This wizard step is displayed only if in the second step of the wizard you selected **Virtual Dimension: The member properties of another dimension**.

Options

Available Dimensions

Click the dimension containing member properties on which the virtual dimension will be based.

Member Properties

View the member properties of the dimension selected in **Available Dimensions**.

Display member keys and names

Click to display all dimensions in the database, including private dimensions. This enables you to use keys and names as if they were member properties.

See Also

Creating Virtual Dimensions

Member Properties

Virtual Dimensions

Select Levels for Virtual Dimension (Dimension Wizard)

In this step of the wizard, you select the member properties on which the virtual dimension's levels will be based.

This wizard step is displayed only if, in the second step of the wizard, you selected **Virtual Dimension: The member properties of another dimension**.

Options

Available member properties

Select the member properties on which to base the levels of the virtual dimension.

Selected virtual levels

View the list of virtual levels based on the member properties selected in **Available member properties**.

Select Advanced Options (Dimension Wizard)

In this step of the wizard, you can select advanced options for your dimension.

This wizard step is not displayed if you selected **Mining Model: A predictable column of an OLAP mining model** in the second step of the wizard.

Options

Options

The following options are available depending on the type of dimension that is being created.

Advanced option	Description	Option available for
Changing dimension	Allows you to add and reorder dimension members without having to reprocess cubes.	Regular dimensions (created from single or multiple tables)*
Custom rollups	Allows you to create aggregate dimension members.	Parent-child dimensions
Members with data	Allows data to be associated with members.	Parent-child dimensions
Ordering and uniqueness of members	Enables member sorting.	Regular dimensions (created from single or multiple tables), parent-child dimensions, or virtual dimensions
Storage mode and member groups	Determines the storage location for dimension members and enables member grouping.	Regular dimensions (created from single or multiple tables)
Writeback	Allows users to add,	Parent-child dimensions

	delete, or modify members in a dimension.	

* This option is automatically enabled for parent-child or virtual dimensions as a property.

Select All

Click to select all available advanced options.

Deselect All

Click to clear all selected advanced options.

Set Changing Property (Dimension Wizard)

In this step of the wizard, you confirm whether to create a changing dimension.

This step is displayed only if you selected **Changing dimension** in the **Select advanced options** step of the wizard.

Options

No, the new dimension is not changing

Select to prevent the new dimension to be created as a changing dimension. This is the default option.

Yes, the new dimension is a changing dimension

Select to confirm that the new dimension is to be created as a changing dimension.

See Also

Changing Dimensions

Set Custom Rollups (Dimension Wizard)

In this step of the wizard, you enable and define custom rollup expressions.

This step is displayed only if you selected **Custom rollups** in the **Select advanced options** step of the wizard.

Options

Enable custom rollups

Select to enable custom rollup expressions for the new dimension.

- **Operator-defined custom rollup**

  Select to use mathematical operators to define the custom rollup.

- **Operator column**

  Type the name of the existing dimension table column that will store the mathematical operators, or click the edit (**...**) button to display the **Define Custom Member Column** dialog box, where you can select an existing column or define a new one.

- **MDX-defined custom formula**

  Select to use a Multidimensional Expressions (MDX) expression to define the custom rollup.

- **Expression column**

  Type the name of the existing dimension table column that will store the MDX expressions, or click the edit (**...**) button to display the **Define Custom Member Column** dialog box, where you can select an existing column or define a new one.

See Also

Define Custom Member Column Dialog Box

Set Members with Data Property (Dimension Wizard)

In this step of the wizard, you can enable associated data for nonleaf dimension members, that is, dimension members with one or more descendants.

This step is displayed only if you selected **Members with data** in the **Select advanced options** step of the wizard.

Options

Nonleaf members have associated data

Select to allow dimension members to have associated data in the fact table.

- **Data members are visible**

  Select to display and aggregate data for nonleaf members as a visible data member.

- **Data members are hidden**

  Select to have data for nonleaf members override the aggregate value of the dimension's regular members.

Specify Ordering and Uniqueness (Dimension Wizard)

In this step of the wizard, you can enable associated data for nonleaf dimension members, that is, dimension members with one or more descendants.

This step is displayed only if you selected **Ordering and uniqueness** in the **Select advanced options** step of the wizard. In addition, available options for this step differ for parent-child dimensions.

Options (Regular and Virtual Dimensions)

Name

View the name of the level.

Order by

Select **<name>** to sort by member name, **<key>** to sort by the member key column, or **<column>** to sort by the column name. If you select **<column>**, the **Select Column** dialog box is displayed.

Keys unique

Select the object that determines the scope of unique member values.

Names unique

Select the object that determines the scope of unique member names.

Options (Parent-Child Dimensions)

Order members by

Select **<name>** to sort by member name, **<key>** to sort by the member key column, or **<column>** to sort by the column name. If you select **<column>**, the **Select Column** dialog box is displayed.

Names are unique among

Select the object that determines the scope of unique member names.

Specify Storage Mode and Member Groups (Dimension Wizard)

In this step of the wizard, you can determine the storage mode for your dimension.

This step is displayed only if you selected **Storage mode and member groups** in the **Select advanced options** step of the wizard.

Options

Store as multidimensional OLAP (MOLAP)

Select to store your dimension's data on the Analysis server. This is the default option.

- **Create member groups for the lowest level**

  Select to automatically create a grouping level if the lowest level of the dimension contains more than 64,000 members.

Store as relational OLAP (ROLAP)

Select to optimize query performance for extremely large dimensions, those generally having 5 million members or more. Dimension data remains in the dimension tables.

- **Enable real-time updates**

  Select for the dimension to support real-time updates.

See Also

Creating Member Groups

Dimension Storage Modes

Real-Time Cubes

Set Dimension Writeback Capability (Dimension Wizard)

In this step of the wizard, you can enable writeback for a new parent-child dimension.

This step is displayed only if you selected **Writeback** in the **Select advanced options** step of the wizard.

Options

Enable writeback in this dimension

Select to enable changes to the dimension to be written directly to the dimension table.

Select Mining Model and Predictable Column (Dimension Wizard)

In this step of the wizard, you select an OLAP data mining model and a predictable column used to create your data mining dimension.

This step is only displayed if you selected **Mining Model: A predictable column of an OLAP mining model** in the second step of the wizard.

Options

Select OLAP mining model

Choose the mining model on which you want to base your dimension.

Select predictable column

Choose the column that will be the source of the dimension's data. Only available predictable columns from the mining model are displayed.

Finish (Dimension Wizard)

In this final step of the wizard, you can name your dimension and preview the dimension data. If you started the Dimension Wizard from Cube Editor or the Cube Wizard, you can also choose whether you want this to be a shared dimension or a private dimension.

Options

Dimension name

Type a unique name for your new dimension. You can enter a maximum of 24 characters for the dimension name, which must begin with an alphabetical character.

Create a hierarchy of a dimension

Select this check box to create a hierarchy and organize the dimension to support multiple hierarchies. For more information, see Creating Dimensions with Multiple Hierarchies.

Hierarchy name

Type a name for the new hierarchy. (This box is available only if you select the **Create a hierarchy of a dimension** check box.) The hierarchy name will be appended to the dimension name, separated by a period to distinguish the hierarchy from other hierarchies in the dimension.

Preview

Preview the hierarchy of your dimension.

Share this dimension with other cubes

Select this option to make this dimension a shared dimension. (This option is available only if you started the Dimension Wizard from Cube Editor or the Cube Wizard. Select it if you want your dimension to be available to other cubes.) If you choose to save this dimension as a shared dimension, the dimension will appear in the Shared Dimensions folder in the Analysis

Manager tree pane.

If you started the Dimension Wizard from the shortcut menu of a Shared Dimensions folder in the Analysis Manager tree pane, your dimension will be shared automatically.

Incremental Update Wizard

Use this wizard to update a partition in your cube with new data from a data source.

Caution This wizard updates a partition. Incorrect use of partitions can result in inaccurate cube data. For more information, see Managing Partitions.

The Incremental Update Wizard appears when, in the Analysis Manager tree pane, you right-click your cube, click **Process**, and then in the **Process a Cube** dialog box, click **Incremental update** and then click **OK**.

The Incremental Update Wizard has the following steps:

- Select the partition to update, if the cube has more than one partition.
- Specify the data source and the fact table.
- Create a filter expression.
- Update the partition in your cube.

Introduction (Incremental Update Wizard)

Use this wizard to add new data to an existing cube partition. The wizard creates a temporary partition from the new data and merges it into an existing partition.

Caution This wizard updates a partition. Incorrect use of partitions can result in inaccurate cube data. For more information, see Managing Partitions.

Running the wizard is appropriate when new data has been added to the data warehouse, but existing data has not changed, and you want to add the new data to your cube. The wizard adds the new data and associated aggregations but does not process changes to the cube's structure.

To start the wizard, in the Analysis Manager tree pane, right-click your cube, and then click **Process**. In the **Process a Cube** dialog box, click **Incremental update**, and then click **OK**.

Options

Skip this screen in the future

Select to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

Select Partition (Incremental Update Wizard)

In this step of the wizard, you select the partition you want to update. This step of the wizard appears only if the cube you are updating contains more than one partition. Because you can update only one partition at a time, you must select the one you want to update.

Options

Partition

Select the partition you want to update.

Specify Data Source and Fact Table (Incremental Update Wizard)

In this step of the wizard, you select the data source and fact table that contain the new data to add to your partition. The data source and fact table already used by the partition are displayed by default. If they also contain the new data you are adding, click **Next**.

You can select a data source and/or fact table different from those already used by the partition.

Options

Data source

> View the data source that contains new data for the incremental update. If you select a different data source, it must contain a fact table with the same structure and columns as the partition's fact table, and it must contain dimension tables with the same structure and columns as the partition's dimension tables. The wizard displays the required dimension tables.

Fact table

> View the fact table that contains the new data for the incremental update. If you select a different table, it must have the same structure and columns as the partition's fact table. You must also manually merge the table with the partition's fact table after the incremental update completes.

Change

> Click this button to open the **Choose a Fact Table** dialog box, where you can select a new data source and fact table.

The required dimension tables are

> View a list of the dimension tables your data source must contain. If you select a different data source, it must contain fact and dimension tables with the same structure and columns as the cube's data source.

See Also

Fact Table Considerations When Merging Partitions

Introduction to Partitions

Managing Partitions

Create Filter Expression (Incremental Update Wizard)

In this step of the wizard, you create an SQL filter expression (WHERE clause expression) to update your partition with only a subset of the data in the fact table.

Caution If in the preceding step of the wizard, you selected the default fact table (that is, the same table already used as the partition's fact table), you must use a filter expression to ensure that only data not already in the partition is added. Otherwise, the cube containing the partition will contain duplicate and therefore inaccurate data.

Options

Create a filter expression

Type a WHERE clause expression. Do not type WHERE. Filters consist of one or more expressions using columns in the fact table. A filter can also contain columns in dimension tables if they are included in a nested SELECT statement and the underlying database supports nested SELECT statements. The filter expression acts as a pass-through statement, and its syntax is not checked until you finish the wizard. If the syntax is incorrect, the incremental update fails.

For example, the partition contains data for years 1995 through 1997. You are adding data for 1998 from the same table that supplies the 1995 through 1997 data. The name of the column that contains years is **the_year**. You must use the following filter expression:

```
"the_year"=1998
```

This example uses alphanumeric data:

```
"the_month"='January'
```

Whenever necessary to avoid ambiguity, use a qualified expression. For example, if a column name appears in multiple tables, include the table name in the expression:

```
"time"."the_month"='January'
```

The SELECT statement used to retrieve records for the incremental update is generated automatically by the wizard. The filter expression is connected with an AND to the automatically generated part of the WHERE clause. Therefore, if you specify multiple filter expressions, enclose them all in a pair of parentheses. For example:

```
("the_year"=1998 OR "the_year"=1999)
```

See Also

Partition Filters and Incremental Update Filters

Finish (Incremental Update Wizard)

In this final step of the wizard, you finish setting options for the incremental update and begin to incorporate the new data into the partition. Depending on the size of the fact table, this operation may take a long time.

Options

Finish

Click to start the incremental update of your partition.

Mining Model Wizard

Use this wizard to create a data mining model. A mining model enables you to analyze your data for patterns and to make predictions based on the patterns. You can create a mining model from a relational schema or a cube, and you can store output from the model in a tabular column, a cube dimension, or a mining model diagram.

The Mining Model Wizard appears when you perform any of the following actions:

- In the Analysis Manager tree pane, right-click the Mining Models folder, and then click **New Mining Model**.
- In the Analysis Manager tree pane, right-click a cube, and then click **New Mining Model**.
- In the Analysis Manager tree pane, select a cube, and then on the **Action** menu, click **New Mining Model**.

After the **Introduction** step, the Mining Model Wizard begins with the following step if you start the wizard by right-clicking the Mining Models folder or by clicking **New Mining Model** on the **Action** menu:

- Choose the type of data source for the mining model. You can create it from:
    - Relational tables. This option creates a relational mining model.
    - A cube. This option creates a multidimensional mining model.

The next steps depend on the type of data source you select. For a relational mining model, the Mining Model Wizard has the following steps:

- Select the table or tables for your mining model.

- Select the data mining technique to be used by your mining model.
- Edit joins. This step is displayed only if you are creating a mining model from multiple tables.
- Select the case key column for your mining model.
- Select the input and predictable columns. You select predictable columns only if you select Microsoft Decision Trees as your data mining technique.
- Finish. Name and save your mining model and optionally process it to view its results.

For OLAP data sources, which are directly specified either by using the wizard or by starting the wizard from selecting a cube, the Mining Model Wizard has the following steps:

- Select the source cube for your mining model. If you began the wizard by right-clicking a cube, this step is not displayed.
- Select the data mining technique to be used by your mining model.
- Select the case dimension and level to be analyzed by the mining model.
- Select the initial predicted entity. This can be a measure of the source cube, a member property of the case level, or members of another dimension. This step appears only if you select Microsoft Decision Trees as your data mining technique.

- Select training data. Training data is known data that is represented by cube elements such as dimensions, levels, member properties, or measures. These are used by the model, the case dimension and the case level to derive the predicted entity by extrapolating from the known training data.

- (Optional.) Create a dimension and/or a virtual cube. The wizard can create a dimension and a virtual cube from the results of the model analysis, enabling you to browse the model, include its cubes, or compare it to the source data. This step appears if you select Microsoft Decision Trees as your data mining technique.

- Finish. Name and save your mining model and optionally process it to view its results.

See Also

Building and Using Data Mining Models

Data Mining Model Structure

Properties Pane (Cube Editor Data View)

Properties Pane (Virtual Cube Editor)

Introduction (Mining Model Wizard)

Use this wizard to create a data mining model.

Options

Skip this screen in the future

Select to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

Select Source Type (Mining Model Wizard)

In this step of the wizard, you select the type of source data from which you want to build your mining model. The data source type you select determines the steps that follow in the wizard.

Options

Relational data

Select to create your mining model from relational tables in data sources supported by Microsoft® SQL Server™ 2000 Analysis Services.

OLAP data

Select to create your mining model from an existing cube. If you select this option, in a later step of this wizard you can create a dimension and a virtual cube to see the results of the data mining analysis.

See Also

Relational Model Steps (Mining Model Wizard)

OLAP Model Steps (Mining Model Wizard)

Relational Model Steps (Mining Model Wizard)

If you specify a relational data source, the Mining Model Wizard has the following steps:

- Select the table or tables for your mining model.
- Select the data mining technique to be used by your mining model.
- Edit joins. This step is displayed only if you are creating a mining model from multiple tables.
- Select the case key column for your mining model.
- Select the input and predictable columns.
- Finish.

Select Case Tables (Mining Model Wizard)

In this step of the wizard, you select the table or tables that contain the case and attribute columns you want to analyze. The case column uniquely represents the entity being analyzed by the mining model, and the attribute columns represent entities that can be predicted by the model.

Options

A single table contains the data

Select the table from the **Available tables** pane by clicking its name or icon.

Multiple tables contain the data

Select tables from the **Available tables** pane.

Available tables

Select a table or tables by clicking table names or icons. Use the buttons provided to move tables to the **Selected tables** pane or remove tables from the **Selected tables** pane.

New Data Source

Click to display the **Data Link Properties** dialog box, where you can specify a new data source for relational tables.

Browse Data

Click to view data in the table selected in **Available tables**.

To specify a data source

Select Data Mining Technique (Mining Model Wizard)

In this step of the wizard, you select the algorithm used to build your mining model.

Options

Select a data mining technique

Choose from two available algorithms.

Option	Description
Microsoft Clustering	Select this option if you want to see general patterns or groupings in your data.
Microsoft Decision Trees	This technique is useful if you want to make specific predictions from the source data. Select this option if you want to create a predictable column in a later step of this wizard.

See Also

Data Mining Algorithms

Create and Edit Joins (Mining Model Wizard)

In this optional step of the wizard, you can change the default table joins that were established when the multiple tables option was previously selected in the wizard.

This wizard step appears only if you selected **Multiple tables contain the data** in the **Select case tables** step of the wizard.

Options

Tables

Select a column from one dimension table and drag to the corresponding column in another dimension table to create a join. To delete a join, right-click the join and then click **Remove**.

Select the Key Column (Mining Model Wizard)

In this step of the wizard, you select the column containing the case key, which is the column in the dimension table that uniquely identifies the case you want to analyze in the mining model.

Options

Case key table

Select the dimension table that contains the case key column. This option is available only if you specified **Multiple tables contain the data** in the **Select case tables** step.

Case key column

Select the column from the dimension table that contains the case key.

See Also

Data Mining Columns

Select Input and Predictable Columns (Mining Model Wizard)

In this step of the wizard, from the available columns, you select at least one input column for your mining model. Input columns represent actual data that is used to train the mining model. You also select at least one predictable column if you selected **Microsoft Decision Trees** in the **Select case tables** step. Predictable columns store predicted output from the mining model that is based on the input columns; they are also used as input columns for the mining model.

Options

Available columns

> Select columns from the tree view. Use the buttons provided to move columns to either the predictable columns pane or the input columns pane or to remove columns from the selection. You cannot use the column you selected in the **Select the key column** step as an input column.

Predictable columns

> View the selected predictable columns. This pane is displayed only if you selected **Microsoft Decision Trees** in the **Select case tables** step.

Input columns

> View the selected input columns.

Finish this mining model in the editor

> Select to bypass selection of input and predictable columns complete the mining model definition in Relational Mining Model Editor. If you select this option, you cannot process the mining model in the last step of the wizard.

See Also

Data Mining Columns

Finish (Mining Model Wizard)

OLAP Model Steps (Mining Model Wizard)

If you specify an OLAP data source, the Mining Model Wizard has the following steps:

- Select the source cube for your mining model. If you started the wizard by right-clicking a cube, this step is not displayed.
- Select the data mining technique to be used by your mining model.
- Select the case dimension and level you want your mining model to analyze.
- Select the predicted entity. This step appears only if you selected **Microsoft Decision Trees** in the **Select data mining technique** step.
- Select training data.
- (Optional.) Create a dimension and/or a virtual cube.
- Finish.

  Note Objects that are not visible in the source cube of an OLAP mining model cannot be included in the mining model. For example, if a dimension has its **Visible** property set to **False** in Cube Editor or Virtual Cube Editor, it cannot be a case dimension. Also, virtual dimensions created in Microsoft® SQL Server™ 7.0 OLAP Services cannot be included in mining models.

See Also

Creating OLAP Data Mining Models

Properties Pane (Cube Editor Data View)

Properties Pane (Virtual Cube Editor)

Select Source Cube (Mining Model Wizard)

In this step of the wizard, you select the cube from the current database that provides the source data for your model.

This wizard step appears only if you started the wizard by right-clicking the Mining Models folder or by clicking **New Mining Model** on the **Action** menu.

Options

Cube

Click a cube to select it.

Dimensions

View the dimensions of the selected cube.

Select Data Mining Technique (Mining Model Wizard)

In this step of the wizard, you select the algorithm used to build your mining model.

Options

Select a data mining technique

Choose from two available algorithms.

Option	Description
Microsoft Clustering	Select this option if you want to see general patterns or groupings in your data.
Microsoft Decision Trees	This technique is useful if you want to make specific predictions from the source data. Select this option if you want to create a dimension from the analysis results or a virtual cube containing the new dimension in a later step of this wizard.

See Also

Data Mining Algorithms

Select Case (Mining Model Wizard)

In this step of the wizard, you select a dimension and, optionally, a level that represent the case, the cube entity that you want to analyze. For example, in a customer credit application, the case is the customer.

Options

Dimension

Select the dimension that represents the case you want to analyze. Only dimensions that have at least one visible level are available. This option selects the dimension along with its lowest dimension by default.

Level

(Optional.) Select a level that further defines the case dimension you want to analyze. This option allows you to select another level other than the dimension's lowest level.

See Also

Creating OLAP Data Mining Models

Select the Predicted Entity (Mining Model Wizard)

In this step of the wizard, you select a predicted entity. This step appears only if you selected **Microsoft Decision Trees** in the **Select data mining technique** step.

After the mining model has been built and processed, you can use OLAP Mining Model Editor to add additional entities to be predicted.

Options

A measure of the source cube

Select an existing measure from the source cube.

A member property of the case level

Select an existing member property of the case level selected in the **Select case** step of the wizard.

Members of another dimension

Select a dimension that differs from the dimension selected in the **Select case** step of the wizard.

See Also

Creating OLAP Data Mining Models

OLAP Mining Model Editor

Select Training Data (Mining Model Wizard)

In this step of the wizard, you select cube elements, such as dimensions, levels, or member properties, that represent actual data with which to train the mining model. These elements are in addition to the case dimension member you selected in the **Select case** step of the wizard. Entities can include dimensions, individual levels with their respective dimensions, or measures.

Options

Cube structure

Select the check box next to a structural element to include it in the training data. By default, all members of the case dimension are included. You can exclude members of the case dimension by clearing corresponding check boxes. However, you must select at least one element in addition to the case dimension.

See Also

Training Data Mining Models

Create a Dimension and Virtual Cube (Mining Model Wizard)

In this optional step of the wizard, you can create a new dimension that contains the results of the mining model analysis. You can also create a virtual cube that contains the created dimension and the model's source cube.

Note If the source cube of the mining model contains a distinct count measure, this feature is unavailable.

Options

Create a new dimension based on this mining model

Select to provide a name for the new dimension.

Dimension name
Type a unique name for your new dimension. You can enter a maximum of 24 characters for the dimension name, which must begin with an alphabetical character.

Create a new virtual cube

Select to create a virtual cube that incorporates both the new dimension and the source cube.

Virtual Cube Name
Type a name unique to the database for the new virtual cube. You can enter a maximum of 50 characters for the virtual cube name. The name must begin with an alphabetical character.

Finish (Mining Model Wizard)

In this final step of the wizard, you name the mining model and can optionally process it.

Options

Model Name

Type a name unique to the database for the new mining model. You can enter a maximum of 50 characters for the name, and the name must begin with an alphabetical character.

Save, but don't process now

Select to postpone mining model processing to a later time. You must process the model before you can view its structure.

Save and process now

Select to process the mining model when you click **Finish**. If this option is unavailable for a relational mining model, you can process the mining model after completing its definition in Relational Mining Model Editor.

Finish

Click to finish the wizard. If you chose the **Save and process now** option, the **Process** dialog box appears in which you can view the progress of the operation.

Partition Wizard

Use this wizard to separate one logical cube into separate physical partitions. This allows you to enhance flexibility in data storage and location of data sources, and improve query performance. The wizard also helps you add partitions to cubes.

CAUTION It is possible to create partitioned cubes that contain incorrect data. For more information, see Managing Partitions.

The Partition Wizard appears when you expand your cube in the Analysis Manager tree pane, right-click the Partitions folder, and then click **New Partition**.

The Partition Wizard has the following steps:

- Specify the data source and fact table for the partition.
- (Optional.) Select a data slice (that is, subset of cube data) to be stored in the partition.
- Specify the partition type: local or remote.
- Final step:
  - Name your partition.
  - Specify filters (WHERE clause expressions) and/or set the aggregation prefix.
  - Start the Storage Design Wizard to design your partition's aggregations (optional) or copy the aggregation design of another partition.

- Save your partition.
- (Optional.) Process your partition.

Note You can create multiple partitions in a cube only if you install Analysis Services for Microsoft® SQL Server™ 2000 Enterprise Edition.

Introduction (Partition Wizard)

Use this wizard to separate one logical cube into separate physical partitions. Using partitions can enhance flexibility in data storage and data source location, and can also improve query performance. The wizard also helps you add partitions to cubes.

CAUTION It is possible to create partitioned cubes that contain incorrect data. For more information, see Managing Partitions.

You can partition data on different servers (using multiple data sources). This approach provides parallel query processing for one cube across a cluster of servers.

You can also separate data into multiple partitions on a single server (using a single data source). This enables you to fine-tune your cube by specifying different data storage modes—multidimensional OLAP (MOLAP), relational OLAP (ROLAP), or hybrid OLAP (HOLAP)—and designing different aggregations for each partition.

To start the Partition Wizard, expand your cube in the Analysis Manager tree pane, right-click the Partitions folder, and then click **New Partition**.

Options

Skip this screen in the future

Select this check box to bypass the wizard's introductory step the next time you start the wizard. If you select this option, you can return to the introductory step in the future by clicking **Back** in the second step of the wizard.

See Also

Introduction to Partitions

Partition Storage

Remote Partitions

Specify Data Source and Fact Table (Partition Wizard)

In this step of the wizard, you select the data source and fact table for your partition. The data source and fact table used by the cube are displayed by default. If they also contain the data for your partition, click **Next**.

You can select a data source and/or fact table different from those used by the cube.

Options

Data source

View the data source your new partition will use.

Fact table

View the fact table your new partition will use.

Change

Click to display the **Choose a Fact Table** dialog box, where you can select a new data source and fact table. If you select a different fact table, it must have the same structure and columns as the cube's fact table.

The required dimension tables are

View a list of the dimension tables your data source must contain. If you select a different data source, it must contain fact and dimension tables with the same structure and columns as the cube's data source.

See Also

Introduction to Partitions

Managing Partitions

Select Data Slice (Partition Wizard)

In this optional step of the wizard, you can specify a subset, or slice, of the data in your cube to use as the partition data. To bypass this step, click **Next**.

Options

Dimensions

View all the dimensions in the cube in the **Name** column, and view the member you select for each dimension in the **Data slice** column. The data slice is defined by all the dimension members you select.

Members

Expand members to reveal other members, and then click the member to define the data slice. To remove a dimension member from the data slice, click the dimension name, and then press DELETE.

Specify Partition Type (Partition Wizard)

In this step of the wizard, you specify whether the partition's data is stored on the local Analysis server or on a remote Analysis server.

You can change the options in this step only if you are creating a new partition. You cannot change them if you are editing an existing partition.

Options

Local

Select to store the partition's data on the local Analysis server (that is, the Analysis server on which the partition is being defined). This is the default option.

Remote

Select to store the partition's data on a remote Analysis server.

If a remote partition is defined on an Analysis server, the Analysis server service (MSSQLServerOLAPService) logon account must be a domain user account. Otherwise, processing the partition will fail. This restriction applies to the Analysis server on which the remote partition is defined, not the Analysis server in the **Select the server this partition will reside on** box.

Select the server this partition will reside on

Select the remote Analysis server on which to store the partition's data. Only registered Analysis servers appear in the box. Remote partitions on a server computer running SQL Server™ 7.0 OLAP Services are not supported.

Register server

Click to display the **Register Analysis Server** dialog box in which you can register an Analysis server. To register an Analysis server, your user name must be included in the OLAP Administrators group on that Analysis server.

See Also

Finish (Partition Wizard)

In this step of the wizard, you can name your partition, choose aggregation design options, and specify advanced settings for your partition. You can also process your new partition when the wizard finishes.

Caution It is possible to create partitioned cubes that contain incorrect data. For more information, see Managing Partitions.

> If in an earlier step you selected the default fact table (that is, the same fact table used by the cube), you must use a filter to ensure that each partition in the cube includes mutually exclusive data. Otherwise, the cube containing the partition will contain duplicate and therefore inaccurate data. To create a filter, click **Advanced**.

Options

Partition Name

Type a name that is unique in the database. You can enter a maximum of 50 characters.

Design the aggregations for your partition now

Select to start the Storage Design Wizard when you click **Finish**.

Design the aggregations later

Select to save your partition now and design aggregations later.

Copy the aggregation design from an existing partition.

Select to copy the aggregation design from another partition to your new partition.

- **Copy from**

  Select the partition from which to copy the aggregation design.

  Note If in the future you might merge the new partition with another, copy the aggregation design of the other partition. To be merged, partitions must have the same structure and aggregation design.

Process the partition when finished

Select if you want to process the new partition when you click **Finish**. This option is not available if you select **Design the aggregations for your partition now**.

Advanced

Click to display the **Advanced Settings** dialog box, where you can add a filter (WHERE clause expression), change the aggregation prefix, or specify drillthrough options.

Finish

Click to apply the settings you have chosen and either save the partition, process and then save the partition, or start the Storage Design Wizard.

Storage Design Wizard

Use this wizard to specify data storage and query performance options for the aggregations on a cube or partition.

The Storage Design Wizard appears when, in the Analysis Manager tree pane, you expand the Cubes folder, right-click an existing cube, and then click **Design Storage**.

This wizard also appears when you expand the Partitions folder for an existing cube, right-click an existing cube partition, and then click **Design Storage**.

The Storage Design Wizard has the following steps:

- Select the type of data storage for your cube or partition: multidimensional OLAP (MOLAP), relational OLAP (ROLAP), or hybrid OLAP (HOLAP).
- Design aggregations for your cube or partition based on aggregation storage size and query performance options.
- Process the aggregations or save and process the aggregations later.

See Also

Cube Storage

Aggregations

Introduction (Storage Design Wizard)

Use this wizard to specify data storage and query performance options for the aggregations on a cube or partition.

Options

Skip this screen in the future

Select this option to bypass this step the next time you start the wizard. If you select this option, you can return to this step in the future by clicking **Back** in the second step of the wizard.

See Also

Cube Storage

Aggregations

Select a Partition (Storage Design Wizard)

This step of the wizard appears only if you started the wizard on a cube and it contains multiple partitions. You can run the wizard on only one partition at a time, so you must select the desired partition.

Options

Partition

Select the partition for which you want to design aggregations and choose storage options.

Aggregations Already Exist (Storage Design Wizard)

This step of the wizard appears only if there are previously designed aggregations for your cube or partition. This step contains information about the existing data storage and aggregation settings.

Options

Data storage type

View the data storage type used in this cube partition.

Aggregation storage space

View the existing storage space used by the previously designed aggregations in this cube partition.

Number of aggregations

View the number of aggregations that have been previously designed for this cube partition.

Replace the existing aggregations

Select this option to completely replace the existing aggregations with new aggregations that you design. To use new aggregations, you must reprocess the cube after aggregation design.

Add new aggregations to the existing ones

Select this option to append new aggregations to the existing ones. You must reprocess the cube before you can use combined aggregations for this cube partition.

Select Data Storage (Storage Design Wizard)

In this step of the wizard, you specify the type of data storage you want to use to store the data and aggregations for a cube or partition.

For more information about data storage choices, see Flexible Data Model.

Options

MOLAP

Select this option to store the data for your cube or partition in a multidimensional structure. The aggregations you design for this storage type will also be stored with the multidimensional data.

Multidimensional OLAP (MOLAP) storage provides the potential for the most rapid query response times, depending only on the percentage and design of the cube's aggregations. In general, MOLAP is more appropriate for cubes with frequent use and the necessity for rapid query response.

ROLAP

Select this option to keep the data for your cube or partition in the existing relational data store. Aggregations designed for relational OLAP (ROLAP) will also be stored in the relational database, rather than in a multidimensional structure.

ROLAP query response is generally slower than that available with MOLAP or HOLAP. A typical use of ROLAP is for large datasets that are infrequently queried, such as less recent historical data.

- **Enable real-time updates**

  Select this to enable the partition to support real-time updates. This option is available only in Analysis Services for SQL Server™ 2000 Enterprise Edition, and only when the data source is SQL Server 2000 Enterprise Edition.

HOLAP

Select this option to keep the data for your cube or partition in the existing relational data store and to keep your aggregations in a multidimensional structure.

For queries that access summary data, hybrid OLAP (HOLAP) is equivalent to MOLAP. Queries that access base data, such as a drilldown to a single fact, must retrieve data from the relational database and will not be as fast as if the base data were stored in the MOLAP structure. Cubes stored as HOLAP are smaller than equivalent MOLAP cubes and respond faster than ROLAP cubes for queries involving summary data. HOLAP storage is generally suitable for cubes that require rapid query response for summaries based on a large amount of base data.

See Also

Flexible Data Model

Partition Storage

Real-Time Cubes

Set Aggregation Options (Storage Design Wizard)

In this step of the wizard, you set options for storage and performance to design the aggregations for the cube or partition.

In designing your cube, you will have to balance the storage needs of the aggregation tables against the speed and performance of the queries. The three approaches to achieving this balance are as follows:

- Set the storage size and let Microsoft® SQL Server™ 2000 Analysis Services determine which aggregations to store. This approach works well when you have limited storage space.
- Set the percentage of performance gain and let the necessary aggregation tables take as much storage space as they need.
- Manually determine the best balance by watching the progress of the **Performance vs. Size** graph.

  Note If you selected the ROLAP data storage option in the previous step of the Storage Design Wizard, you can use ROLAP data storage without creating aggregations in the relational data store. To do this, select **Until I click Stop**, and then click **Next** to advance to the **Finish** step of the wizard.

Options

Select the option you want for aggregation design.

Estimated storage reaches

Enter the amount of hard disk storage to allocate for storing the aggregation tables. You can enter a maximum storage size in either megabytes (MB) or gigabytes (GB).

Performance gain reaches

Specify the percentage amount of performance gain for your queries. This amount represents the percentage improvement between the maximum and minimum query times, as represented by the following formula:

PercentGain = 100 * (QTimeMAX - QTimeTARGET) / (QTimeMAX - QTimeMIN)

For example, if a query that is not optimized takes twenty-two seconds (QTimeMAX) to execute, and the best possible query performance with maximum aggregations is two seconds (QTimeMIN), specify a 75% desired performance gain to achieve a query time of seven seconds (QTimeTARGET).

Until I click Stop

Select to manually control the balance. Watch the **Performance vs. Size** graph to determine when the increase in performance levels off, even though storage continues to build.

Start

Click to begin designing aggregations based on the options you have selected.

Continue

Click to resume designing aggregations based on the options you have selected. The **Continue** button replaces the **Start** button after you click **Stop** or the line in the **Performance vs. Size** graph reaches the specified storage or performance gain.

Stop

Click to manually end the aggregation design process.

Reset

Click to delete any aggregations you have just added and restart aggregation design.

Performance vs. Size

View the progress of the design process, including the estimated performance gain and the estimated storage space requirements.

See Also

Partition Storage

Aggregations

Finish (Storage Design Wizard)

In this step of the wizard, you save the aggregations. You can process the cube as you save the aggregations or process it at a later time. If you are designing several cubes, you can postpone processing and process them all at once.

Options

Process now

Select this option to save the designed aggregations for the cube or partition and process the cube. The new aggregations will be available for use by queries sent to the cube only after processing is completed.

Save, but don't process now

Select this option if you want to save the newly designed aggregations for the cube or partition, but you want to delay cube processing to a later time. Your new aggregations will not be available for use by queries sent to the cube until processing is completed.

Finish

Click to save your settings. If you selected **Process now**, clicking **Finish** also displays the **Process** dialog box, in which you can view the progress of the operation.

See Also

Processing Cubes

Usage Analysis Wizard

Use this wizard to generate an on-screen report analyzing the query usage of your cube.

The Usage Analysis Wizard appears when you right-click a cube in the Analysis Manager tree pane and then click **Usage Analysis**.

The Usage Analysis Wizard has the following steps:

- Select a report type.
- (Optional.) Select the criteria to filter your report.
- View the results.

Introduction (Usage Analysis Wizard)

Use this wizard to set criteria to analyze the queries sent to your cube. You can choose from a variety of reports to analyze. You can specify the following report types:

Report type	**Shows**
Table	Length of time a query takes to return a result set
	Number of times a query has been run
	Users and number of queries they have sent
Graph	Response times for all queries
	Number of queries processed each hour
	Number of queries processed in a specified date range

Options

Select a report type, and then click Next

Select the type of report you want the wizard to generate.

Description

View a description of the report type you select.

Select Filter Criteria (Usage Analysis Wizard)

In this optional step of the wizard, you set criteria to analyze query usage. A query log contains information about the queries sent to your cube. To determine the specifics of your report, you apply filters to this log. Available filter criteria include time period, users, query duration, and query frequency.

This step is optional. If you select no filter criteria, the wizard will return a report on all queries. If you want to skip this step, click **Next**.

Options

Queries for the dates

Select to set the one week period of dates you want the wizard to analyze. To choose the start date for the week to be analyzed, ether type the date directly into the **One week beginning with** field, or select the expand (**...**) button to choose the start date. If, in the Introduction step, you selected **Query by Date Graph**, this check box is automatically selected.

Queries by these users

Select to add users and groups of users whose queries you want the wizard to analyze. Because the available users and groups are defined based on user roles, you cannot filter by users unless you have defined user roles. For more information about roles, see Database, Cube, and Mining Model Roles.

Add

Click to display the **Select Users** dialog box, where you can select users from the list of available users.

Remove

Click to remove a selected user or group from the list.

Queries that ran more than

Select to tell the wizard to analyze only queries that were sent more than a certain number of times.

Queries that took longer than

Select to tell the wizard to analyze only queries that took longer than a certain amount of time. Use the two boxes provided to specify the number of seconds or minutes.

Review Results (Usage Analysis Wizard)

In this final step of the wizard, you can view your report. It cannot be saved. The title on this page reflects the report choice you made in the first step of the wizard.

Options

Delete Records

Click to delete queries analyzed for your report from the query log.

Finish

Click to close the wizard.

Usage-Based Optimization Wizard

Use this wizard to tune performance of cube partitions based on the history of queries previously sent to the cube.

For more information about optimizing performance, see Scalability.

The Usage-Based Optimization Wizard appears when, in the Analysis Manager tree pane, you right-click an existing cube and then click **Usage-Based Optimization**.

The Usage-Based Optimization Wizard has the following steps:

- Select the cube partition whose query performance you want to optimize.
- Select the queries on which you want to base your optimization.
- Review the results.
- Choose whether to replace existing aggregations or to add new ones.
- Select a type of data storage.
- Design new aggregations.
- Save and process the new aggregations.

Introduction (Usage-Based Optimization Wizard)

Use this wizard to optimize the aggregations for your cube partitions based on the queries that have been previously sent to your cube.

To start the Usage-Based Optimization Wizard, in the Analysis Manager tree pane, right-click an existing cube, and then click **Usage-Based Optimization**.

Options

Skip this screen in the future

Select this option to bypass the wizard introductory screen the next time you start the wizard. If you select this option, you can return to the introductory screen in the future by clicking **Back** in the second step of the wizard.

Select Partition (Usage-Based Optimization Wizard)

In this step of the wizard, you select the partition whose performance you want to optimize. You can perform this operation on only one partition at a time. This step of the wizard appears only if your cube has multiple partitions.

Options

Partition

Select a partition from the list provided.

Select Queries (Usage-Based Optimization Wizard)

In this step of the wizard, you select the queries on which you want to base your optimization. You can specify:

- Queries between two dates
- Queries by specific users
- Length of time that a query takes to return a result set
- Numbers of times a query has been run
- Queries sent to one of the specific data storage types used in your cube
- A combination of any number of these

Options

Queries for the dates

Select to set the date range for queries on which you want the wizard to base optimization. To choose the date range type, use the drop-down list to select **Between**, **Before**, or **After**. The options available under this check box change depending on the date range type.

- **Between**

  To use all queries between a desired start date and end date for optimization, enter the desired start date in the following text box, or click the expand (**...**) button to select the desired start date. Then, enter the desired end date in the last text box, or click the expand (**...**) button to select the desired end date.

- **Before**

  To use all queries on or before a desired date for optimization, enter the desired date in the following text box, or click the expand (**...**) button to select the desired date.

- **After**

  To use all queries after a desired date for optimization, enter the desired date in the following text box, or click the expand (**...**) button to select the desired date.

Queries by these users

Select to add users and groups of users whose queries you want the wizard to base optimization on. Because the available users and groups are defined based on user roles, you cannot filter by users unless you have defined user roles. For more information about roles, see Database, Cube, and Mining Model Roles.

- **Add**

  Click to display the **Select Users** dialog box, which you can use to select users from the list of available users.

- **Remove**

  Click to remove a selected user or group from the list.

Queries that ran more than

Select to tell the wizard to optimize based only on queries that were sent more than a certain number of times. Choose the number in the box provided.

Queries that took longer than

Select to tell the wizard to optimize based only on queries that took longer than a certain amount of time. Use the two boxes provided to specify the number of seconds or minutes.

Queries to

Select to tell the wizard to optimize performance based only on queries to a certain type of data storage object.

- **MOLAP cubes**

  Select to optimize aggregations based on queries to multidimensional OLAP (MOLAP) cubes.

- **ROLAP tables**

  Select to optimize aggregations based on queries to relational OLAP (ROLAP) tables.

- **The server cache**

  Select to optimize aggregations based on lookup queries to the server cache.

Review Results (Usage-Based Optimization Wizard)

In this step of the wizard, you can view the queries that match the options you have specified for optimization of your cube partition. A status bar near the bottom of the screen displays the number of matching queries found. These results will be used to optimize future queries by modifying existing aggregations.

Options

Request Data Set Description

View information on the data requested by each matching query.

Times Executed

View the number of times each matching query had been sent to the cube.

Average Duration (min:sec)

View the average duration of each matching query.

Aggregations Already Exist (Usage-Based Optimization Wizard)

This step of the wizard appears only if there are previously designed aggregations for your cube or partition. Information appears on the screen about the existing data storage and aggregation settings.

Options

Data storage type

View the data storage type used in this cube partition.

Aggregation storage space

View the existing storage space used by the previously designed aggregations in this cube partition.

Number of aggregations

View the number of aggregations previously designed for this cube partition.

Replace the existing aggregations

Select this option to completely replace the existing aggregations with new ones you design. Before you can use your new aggregations, you must reprocess the cube.

Add new aggregations to the existing ones

Select this option to append new aggregations to the existing ones. You must reprocess the cube to use combined aggregations for this cube partition.

Select Data Storage (Usage-Based Optimization Wizard)

In this step of the wizard, you specify the type of data storage you want to use to store the data and aggregations for your cube or partition. This step appears if you selected **Replace the existing aggregations** in the previous step of the wizard.

Options

MOLAP

Select to store the data for your cube or partition in a multidimensional structure. The aggregations you design for this storage type will also be stored with the multidimensional data.

Multidimensional OLAP (MOLAP) storage provides the potential for the most rapid query response times, depending only on the percentage and design of the cube's aggregations. In general, MOLAP is more appropriate for cubes that are used frequently and which require rapid query response.

ROLAP

Select to keep the data for your cube or partition in the existing relational data store. The aggregations you design for this storage type will also be stored in the relational database, rather than in a multidimensional structure.

Relational OLAP (ROLAP) query response is generally slower than that available with the other two options. A typical use of ROLAP is for large datasets that are infrequently queried, such as historical data from less recent years.

HOLAP

Select to keep the data for your cube or partition in the existing relational data store and your aggregations in a multidimensional structure.

For queries that access summary data, hybrid OLAP (HOLAP) is the equivalent of MOLAP. Queries that access base data, such as a drill-down to a single fact, must retrieve data from the relational database and will not be as fast as if the base data were stored in the MOLAP structure. Cubes stored as HOLAP are smaller than equivalent MOLAP cubes and respond faster than ROLAP cubes for queries involving summary data. HOLAP storage is generally suitable for cubes that require rapid query response for summaries based on a large amount of base data.

See Also

Aggregations

Cube Storage

Partition Storage

Set Aggregation Options (Usage-Based Optimization Wizard)

In this step of the wizard, you set options for storage and performance to design the aggregations for your cube or partition.

In designing your cube, you will have to balance the storage needs of your aggregation tables against the speed and performance of your queries. There are three approaches to achieving this balance:

- Set the storage size and let Microsoft® SQL Server™ 2000 Analysis Services determine which aggregations to store. This approach works well when you have limited storage space.
- Set the percentage of performance gain you want and let the necessary aggregation tables take as much storage space as they need.
- Manually determine the best balance by watching the progress of the **Performance vs. Size** graph.

  Note If you selected the ROLAP data storage option in the previous step of this wizard, you can use ROLAP data storage without creating aggregations in the relational data store. To do this, select **Until I click Stop** and then click **Next** to advance to the Finish step.

Options

Estimated storage reaches

Enter the amount of hard disk storage you want to allocate for storing the aggregation tables. You can enter a maximum storage size in either megabytes (MB) or gigabytes (GB).

Performance gain reaches

Specify the percentage amount of performance gain for your queries. This amount represents the percentage improvement between the maximum and minimum query times, as represented by the following formula:

> PercentGain = 100 * (QTimeMAX - QTimeTARGET) / (QTimeMAX - QTimeMIN)

For example, if an unoptimized query takes twenty-two seconds (QTimeMAX) to execute, and the best possible query performance with maximum aggregations is two seconds (QTimeMIN), specify a 75% desired performance gain to achieve a query time of seven seconds (QTimeTARGET).

Until I click Stop

Select to manually control the balance. Watch the **Performance vs. Size** graph to determine when the increase in performance levels off even though storage continues to build.

Start

Click to begin designing aggregations based on the options you have selected.

Continue

Click **Continue** to resume designing aggregations based on the options you have selected. The **Continue** button replaces the **Start** button after you click **Stop** or the line in the **Performance vs. Size** graph reaches the specified storage or performance gain.

Stop

Click to manually halt the aggregation design process.

Reset

Click to delete any aggregations you have just added and restart aggregation design.

Performance vs. Size

View the progress of the design process, including the estimated performance gain and the estimated storage space requirements.

Finish (Usage-Based Optimization Wizard)

In this step of the wizard, you save the aggregations. You can process the cube while you save the aggregations, or process it at a later time. If you are designing several cubes, you can postpone processing and process them all at once.

Options

Process now

Select to save the designed aggregations for the cube or partition and process the cube. The new aggregations will be available for use by queries sent to the cube only after processing is completed.

Save, but don't process now

Select to save the newly-designed aggregations for the cube or partition, but to delay cube processing to a later time. Your queries sent to the cube cannot use your new aggregations until processing is completed.

Finish

Click to save your settings and, if you selected **Process now**, process the object. If you selected **Process now**, clicking **Finish** opens the **Process** dialog box, in which you can view the progress of the operation.

See Also

Processing Cubes

Virtual Cube Wizard

Use this wizard to create a virtual cube, which is a superset of selected cubes in your database. Virtual cubes allow you to create a broader view of your multidimensional data without storing additional data in a physical storage space on your drive.

The Virtual Cube Wizard appears when, in the Analysis Manager tree pane, you expand your database, right-click the Cubes folder, and then click **New Virtual Cube**.

The Virtual Cube Wizard has the following steps:

- Select the cubes in your database that you want to include in your virtual cube.
- Select the measures for your virtual cube.
- Select the dimensions for your virtual cube.
- Name and save your virtual cube.

See Also

Virtual Cubes

Introduction (Virtual Cube Wizard)

Use this wizard to create a virtual cube. Select the cubes that you want to include in the virtual cube, and then select the specific measures and dimensions that you want to use in the virtual cube.

Options

Skip this screen in the future

Select to bypass the wizard introductory screen the next time you start the wizard. If you select this option, you can return to the introductory screen in the future by clicking **Back** in the second step of the wizard.

Select Cubes (Virtual Cube Wizard)

In this step of the wizard, you select the cubes from your database that you want to add to your virtual cube.

Options

Available cubes

Use the buttons provided to move cubes to the **Virtual cube includes** box.

Virtual cube includes

View the list of cubes you have selected to include in your virtual cube.

See Also

Virtual Cubes

Select Measures (Virtual Cube Wizard)

In this step of the wizard, you add measures to your virtual cube.

Options

Available measures

Use the buttons provided to move cubes to the **Virtual cube includes** box.

Selected measures

View the list of measures you have selected to include in your virtual cube.

See Also

Virtual Cubes

Select Dimensions (Virtual Cube Wizard)

In this step of the wizard, you add dimensions to your virtual cube.

The dimensions that are available for your virtual cube are listed in the **Available dimensions** box. Dimension availability is based on the measures you have selected for your virtual cube. Only the dimensions from cubes that contain your selected measures are available to your virtual cube. If the **Available dimensions** box does not contain a dimension that you want to add, make certain that you have selected a measure from the cube containing that dimension.

Options

Available dimensions

Use the buttons provided to move cubes to the **Selected dimensions** box.

Selected dimensions

View the list of dimensions you have selected to include in your virtual cube.

See Also

Virtual Cubes

Finish (Virtual Cube Wizard)

In this final step of the wizard, you name the virtual cube.

Options

Virtual cube name

Type a name unique to the database for the new virtual cube. You can enter a maximum of 50 characters for the virtual cube name. The name must begin with an alphabetical character.

Process now

Select this option to process the virtual cube when you click **Finish**. If the structure of a component cube has changed since the last time it was processed, the cube is processed along with the virtual cube; component cube processing may take considerable time.

Save and don't process now

Select this option to postpone virtual cube processing to a later time. You must process the virtual cube before it can be used.

Finish

Click to save your settings and optionally process the cube. If you selected **Process now**, clicking **Finish** displays the **Process** dialog box, where you can view the progress of the operation.

See Also

Virtual Cubes

Analysis Services

Dialog Boxes

This section contains Help topics that are available from the user interface by pressing F1 or by clicking **Help** in dialog boxes.

See Also

Wizards

Advanced Settings Dialog Box

Use this dialog box to:

- Create a filter (WHERE clause expression) that limits the data used to create the partition.

- Change the prefix for the aggregation names.

- Access the **Drillthrough Options** dialog box, where you can specify drillthrough options for the partition.

  CAUTION It is possible to create partitioned cubes that contain incorrect data. For more information, see Managing Partitions.

This dialog box is displayed when you click **Advanced** in the **Convert to Partition** dialog box or in the last step of the Partition Wizard.

Options

Filter statement

Type a filter expression (WHERE clause expression of an SQL SELECT statement) to limit the data used to create the partition. Do not type WHERE.

IMPORTANT Filter expressions can be used to ensure that all partitions in a cube contain mutually exclusive data. Otherwise, the cube will contain duplicate and therefore inaccurate data. When you use a filter expression for this purpose, make sure that it excludes data already in the partitions of the cube.

For example, a cube contains three partitions, one each for the years 1995, 1996, and 1997. You are creating a new partition for 1998 from the same table that supplies the data for the other partitions. The name of the column that contains years is **the_year**. You must use the following filter expression:

```
"the_year"=1998
```

This example uses alphanumeric data:

```
"the_month"='January'
```

If necessary, use a qualified expression to avoid ambiguity. For example, if a column name appears in multiple tables, include the table name in the expression:

```
"time"."the_month"='January'
```

The SELECT statement used to retrieve records for the creation of the partition is generated automatically. The filter expression is connected with an AND to the automatically generated part of the WHERE clause. Therefore, if you specify multiple filter expressions, you must enclose them all in a pair of parentheses. For example:

```
("the_year"=1998 OR "the_year"=1999)
```

Filters consist of one or more expressions using columns in the fact table. A filter can also contain columns in dimension tables if they are included in a nested SELECT statement and the underlying database supports nested SELECT statements. For more information, see Partition Filters and Incremental Update Filters.

The filter expression acts as a pass-through statement, and its syntax is not checked until you process the partition. If the syntax is incorrect, processing fails.

Aggregation Prefix

Specify the prefix used for the aggregation names of the partition.

Drillthrough Options

Click to display the **Drillthrough Options** dialog box, where you can specify drillthrough options for the partition. For more information, see Drillthrough Options Dialog Box.

Analysis Services Processing Task Dialog Box

Use this dialog box to add a Data Transformation Services (DTS) task that performs processing of one or more objects defined in Microsoft® SQL Server™ 2000 Analysis Services. For more information about using DTS tasks to automate processing, see Processing Objects Using Data Transformation Services.

This dialog box is displayed when you perform either of the following actions in DTS Designer:

- Drag the icon for the Analysis Services Processing task from the **Task** toolbar to the design sheet.

- Right-click an Analysis Services Processing task, and then click **Properties**.

Options

Name

View the task name.

Description

Specify a description for the task (optional).

Select the object to process

Select an object or folder to process.

Note If Analysis Manager is not installed on the system, you will be prompted to enter the name of an Analysis server. This is the name of the server, not its IP address.

Object	Object icon	Processing includes
Database	🗄	All cubes, partitions, virtual cubes, and dimensions in the

		database.
Cubes folder		All cubes and virtual cubes in the folder.
Cube		The selected cube.
Partition		The selected partition.
Remote partition		The selected remote partition.
Linked cube		The selected linked cube.
Virtual cube		The selected virtual cube.
Dimensions folder		All dimensions in the folder.
Shared dimension		The selected shared dimension.
Virtual dimension		The selected virtual dimension.
Relational mining model		The selected relational mining model.
OLAP mining model		The selected OLAP mining model.

Local server

Select this option to:

- Limit the objects in the **Select the object to process** box to those on the local Analysis server.

- Specify that the task processes an object or objects on the server computer where the package is stored.

  If you select this option, you can later migrate the package to another instance of SQL Server and execute it to process an object or objects on that instance of SQL Server. This processing requires that the other instance of SQL Server stores the meta data for the processed objects and has access to their data sources. Meta data can be copied easily from one server to another by copying and pasting.

Select a processing option

Select a processing option for the object or folder you selected. The available options change according to the object type or folder you select.

Some objects and folders have only one available option. For more information about those with multiple options, see Processing Cubes, Incremental Updates and Partitions, and Dimension Processing.

Data source

View the data source for an incremental update of a cube or partition. You can change the data source by clicking the edit (**...**) button beside the **Fact table** box and using the **Choose a Fact Table** dialog box.

This option is displayed only when you select a single-partition cube or partition and **Incremental update**.

Fact table

View the fact table for an incremental update of a cube or partition. You can change the fact table by clicking the edit (**...**) button beside the **Fact table** box and using the **Choose a Fact Table** dialog box.

This option is displayed only if you select a single-partition cube or partition and **Incremental update**.

Filter

View the filter for an incremental update of a cube or partition. A filter limits the fact table records used in the incremental update. You can add or change a filter by clicking the edit (**...**) button beside the **Filter** box and using the **Filter Expression** dialog box.

This option is displayed only if you select a single-partition cube or partition and **Incremental update**.

Note You must specify a filter if you selected a single-partition cube or partition, **Incremental update**, and the default fact table. Otherwise, the cube or partition will contain duplicate and therefore inaccurate data.

OK

Click to add the task with the values you selected to the package and close the dialog box. (To save the task, you must save the package in DTS Designer.)

Cancel

Click to add the task with default values to the package and close the dialog box.

Archive Database Dialog Box

Use this dialog box to:

- Specify options for archiving a Microsoft® SQL Server™ 2000 Analysis Services database.
- Begin the archive process, which produces an archive file.

The archive file includes files for the selected database that are in a directory with the same name as the database. This directory is in the Data directory, which contains a directory for each database on the server. For example, if you archive the **FoodMart 2000** database, the files in C:\Program Files\Microsoft Analysis Services\Data\FoodMart 2000\ are included in the archive file.

IMPORTANT The subdirectories of the Data directory store the security files that control end users' access to objects on the Analysis server. These files are included in the archive files. For this reason, archive files must be secured against unauthorized access.

The archive file also stores meta data for the database and its objects. The appropriate records from the Analysis Services repository are included in the archive file. By default, the Analysis Services repository is Msmdrep.mdb, but it can be migrated to a SQL Server database.

For more information about archiving Analysis Services databases, see Archiving and Restoring Databases.

The dialog box is displayed when, in the Analysis Manager tree pane, you right-click a database and then click **Archive Database**.

Options

Save in

Specify the name and path of the archive file. To browse available paths, click the browse (**...**) button.

Temp Folder

Specify the directory in which to store temporary files during the archive process. To browse available paths, click the browse (**...**) button.

Remote Path

Specify the path of the Data directory that contains the files for the database you are archiving. To browse available paths, click the expand (**...**) button.

The Data directory is created during installation of Analysis Services and contains a directory for each database on the Analysis server. The default path of the Data directory is C:\Program Files\Microsoft Analysis Services\Data\

The C:\Program Files\Microsoft Analysis Services\ portion of the path can be changed. (To determine the current value, right-click the remote server, click **Properties**, and then view the **Data folder** box.)

In the **Remote Path** box, you must precede the path of the Data directory with the remote server name. If the Data directory is shared at its disk level, you must also include the share name associated with the disk. The following example **Remote Path** value includes the server name Server-1 and the disk share name C$.

```
\\Server-1\C$\Program Files\Microsoft Analysis Services\Data\
```

The **Remote Path** box appears only if the database you are archiving is on a remote server.

Archive

Begin the archive process. The **Archive Database Progress** dialog box is displayed, and it allows you to monitor or cancel the archive process.

Archive Database Progress Dialog Box

Use this dialog box to monitor the archiving of a Microsoft® SQL Server™ 2000 Analysis Services database. You can also cancel the archive process or save the archive log.

For more information about archiving Analysis Services databases, see Archiving and Restoring Databases and Archive Database Dialog Box.

This dialog box is displayed when you click **Archive** in the **Archive Database** dialog box.

Options

Save Log

Click to display the **Save Log File** dialog box, in which you specify the path and file name of the archive log and then save it. If the archive process completes successfully, the archive log contains a list of the files contained in the archive file. Otherwise, the archive log contains one or more messages. The **Save Log** button is available only after the archive process is completed or canceled.

Cancel

Click to cancel the archive process. If you click **Cancel** and receive a message indicating that the archive was canceled, nothing has been archived. A delay may occur between clicking **Cancel** and display of the message.

After the archive process is completed or canceled, the **Cancel** button is replaced by the **Close** button.

Close

Click to close the dialog box. The **Close** button is available only after the archive process is completed or canceled.

Calculated Member Builder

Use this dialog box to add a calculated member to a cube or virtual cube.

This dialog box appears for a cube or a virtual cube when you do one of the following in Cube Editor or in Virtual Cube Editor:

- On the **Insert** menu, click **Calculated Member**.
- In the tree pane, right-click the Calculated Members folder or a calculated member, and then click **New Calculated Member**.
- In the tree pane, right-click a calculated member, and then click **Edit**.
- On the toolbar, click **Insert Calculated Member**.

Options

Parent dimension

Select the dimension in which to create the calculated member.

Parent member

Enter an expression for the parent member of the calculated member, or click **Change** to select a parent member. The parent member determines the location of the calculated member in the dimension structure. This option is enabled if you select a parent dimension (other than the Measures dimension) that has more than one level.

- **Change**

  Click to display the **Select the Parent Member** dialog box. This option is enabled if you select a parent dimension other than Measures that has more than one level.

Member name

Enter a name for the calculated member (for example, Average Sale.)

Value expression

Build or enter the expression that determines the values of the calculated member. To add a function to the **Value expression** box, place the cursor where you want to insert the function, select a function in the **Functions** box, and either double-click the function or click **Insert**. The function syntax appears in the **Value expression** box. Replace arguments and their delimiters (« and ») with the appropriate values.

You can also type an expression directly into the **Value expression** box. You must use this method if you want to add functions from libraries other than the Microsoft® SQL Server™ 2000 Analysis Services Multidimensional Expressions (MDX) function library.

Check

Click to validate the syntax in the **Value expression** box.

Data

Select the cube's dimensions, measures, and existing calculated members to build the expression in the **Value expression** box. To add objects, you can double-click the selected object, select the object and then click **Insert**, or drag the selected object to the **Value expression** box.

Functions

Select a function from the list to build the expression in the **Value expression** box. To add functions, you can double-click the selected object, select the object and then click **Insert**, or drag the selected object to the **Value expression** box. The list includes the Analysis Services MDX function library and any additional function libraries you have registered. For MDX, a short description of the function and an example of the syntax are displayed below the **Functions** and **Data** boxes. For more information about the functions, see MDX Function List.

Insert

Click to add the item selected in the **Data** box or **Functions** box to the value expression.

Number and arithmetic operator buttons

Click to add numbers and operators to the value expression.

Register

Click to display the **Register Function Libraries** dialog box, where you can register an external function type library (*.olb, *.tlb, *.dll). The new library appears in the **Functions** box. For more information, see Register Function Libraries Dialog Box.

See Also

Registered Function Libraries

Select the Parent Member Dialog Box

Choose a Dimension Table Dialog Box

Use this dialog box to choose a dimension table from a selected data source.

This dialog box appears when you create a new dimension in Dimension Editor.

Options

Tables

View the tables of the selected data source.

Details

View a description of the data source if a data source is selected in the **Tables** box. If a table is selected in the **Tables** box, you can view the columns in that table.

New Data Source

Click to display the **Data Link Properties** dialog box, where you can specify a new data source.

See Also

Dimension Editor - Schema View

Choose a Fact Table Dialog Box

Use this dialog box to choose a fact table from a selected data source.

This dialog box appears when you create a new cube in Cube Editor or edit the fact table from the **Analysis Services Processing Task** dialog box.

Options

Tables

View the tables of the selected data source.

Details

View a description of the data source, if a data source is selected in the **Tables** box. If a table is selected in the **Tables** box, you can view the columns of that table.

New Data Source

Click to display the **Data Link Properties** dialog box, where you can specify a new data source. This option is available only if you are creating a new cube in Cube Editor.

Convert to Partition Dialog Box

Use this dialog box to convert a writebacktable to a partition. This action also makes read/write permissions on the cube unavailable. (Users with unavailable read/write permissions can still browse the cube.) Read permissions are not affected.

Caution It is possible to create partitioned cubes that contain incorrect data. For more information, see Managing Partitions.

This dialog box appears when you right-click a cube that has writeback enabled, point to **Writeback Options**, and then click **Convert to Partition**.

Options

Partition name

Type a name for the new partition.

Design the aggregations for your partition now

Start the Storage Design Wizard when you click **OK**.

Design the aggregations later

Save the partition when you click **OK**, but design aggregations later.

Copy the aggregation design from an existing partition

Copy the aggregation design from another partition to the new partition. In the **Copy from** box, select the partition from which to copy existing aggregations.

If in the future you might merge the new partition with another, copy the aggregation design of the other partition. Merged partitions must have the same aggregation design.

Process the partition when finished

Process the new partition when you click **OK**. This option is not available if you select **Design the aggregations for your partition now**.

Advanced

Use to add a filter, to change the aggregation prefix, or to specify drillthrough options. The **Advanced Settings** dialog box is displayed. For more information, see Advanced Settings Dialog Box.

OK

Save changes you have made, convert the writeback table to a partition, and close the dialog box.

Cube Browser

Use this tool to view data in a cube.

Cube Browser organizes the cube data in the data viewing pane. The default view in the data viewing pane shows the data in table format with one dimension across the column headings and another dimension down the left column. Measures are treated as a single dimension for this purpose. The remaining dimensions of the cube are displayed in the data slicing pane.

The white cells in the data viewing pane represent the measure values according to the members that appear in the members boxes for all the dimensions shown in the data slicing pane. In the preceding example, the measure values reflect the All member of every dimension in the data slicing pane except Time. The members box for the Time dimension displays 1997, which indicates that the displayed measure values are limited to 1997. Later in this topic, an example shows you how to slice through dimensions in the data slicing pane to limit the measure values that are displayed.

Cube Browser is displayed when you perform one of the following actions:

- In the Analysis Manager tree pane, right-click a processed cube, and then click **Browse Data**. If the structure of a cube has changed since it was last processed, the cube must be processed before you perform this action. Also, you can view sample data in the **Data** tab of Cube Editor.

- In the last step of the Cube Wizard, click **Browse Sample Data**. The displayed data is sample data rather than actual data because the cube has not yet been processed.

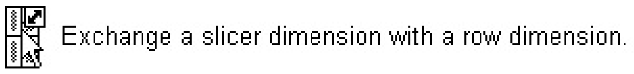
Working in Cube Browser

To browse the cube data in the data viewing pane by any combination of dimensions, drag measures or dimensions from the data slicing pane onto either axis in the data viewing pane.

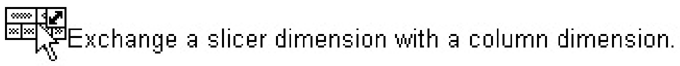
To replace a dimension in the data viewing pane with another dimension

- Drag the dimension from the data slicing pane to the dimension in the data viewing pane that you want to replace.

The pointer is in the correct position when it appears like this:


Exchange a slicer dimension with a row dimension.

-or-

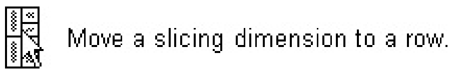

Exchange a slicer dimension with a column dimension.

The dimension in the data viewing pane moves up to the data slicing pane and the new one takes its place.

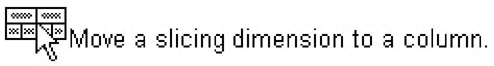
To move a dimension to the data viewing pane

- Drag the dimension from the data slicing pane to a cell below (or beside) the existing dimension.

The pointer is in the correct position when it appears like this:


Move a slicing dimension to a row.

-or-


Move a slicing dimension to a column.

The new dimension appears under (or beside) the existing dimension, expanding your data viewing pane.

To change the order of dimensions in the data viewing pane

- Drag the dimension and drop it on the dimension where you want it to be located.

To remove a dimension from the data viewing pane

- Drag the dimension you want to remove from the data viewing pane to the data slicing pane.

To drill down into a member

- Double-click a dimension member in the data viewing pane. If the member has members beneath it, they are displayed.

  Only the member you double-click is expanded. The other members retain their current levels in the data viewing pane. In the following example, double-click the Drink member to drill down into its child members.

If you want to display all the members in a level, double-click the parent level name in the data viewing pane. Level names are depicted with the shading of a button (for example, Product Family in the preceding example).

To slice through a dimension

- To change the member in a dimension in the data slicing pane, click the down arrow on the members box, expand the members, and then select a new member. The cube data in the data viewing pane changes to reflect the change in the dimension member.

Cube Browser has an internal memory limit that you may reach if you attempt to browse too much data or drill down too deeply. When you reach the limit, the following message is displayed:

```
Unable to display current view of cube.
Unable to Allocate Memory For Flexgrid.
```

The limit cannot be increased by adding or allocating more memory. If you reach the limit, reduce the amount or depth of data you are attempting to browse or use another browser.

Cube Cell Security Dialog Box

Use this dialog box to define a custom rule for cell security.

In this dialog box, there are two ways to allow or deny access to specific cells or groups of cells. You can type an expression in Multidimensional Expressions (MDX) to allow or deny access to specific cells or groups of cells. The MDX expression resolves to either TRUE or FALSE for each cell (atomic and nonatomic) in the cube. (If the MDX expression resolves to a numeric value, any nonzero value is evaluated TRUE, and zero is evaluated FALSE.) If the expression resolves to TRUE for a cell, access is allowed. If it resolves to FALSE, access is denied.

You can also click **Build** to display MDX Builder, in which you can construct the MDX expression by using drag-and-drop techniques.

This dialog box is displayed when, in the **Cells** tab of the **Create a Cube Role** or **Edit a Cube Role** dialog box, you select **Advanced** in the **Cell Security** box, and then select a permission in the **Advanced Cell Security** box. Next, select **Custom** in the **Rule** column, and then click the edit (**...**) button in the **Custom Settings** column.

Options

Permission

View the permission to which the custom rule applies. This box is read-only.

Description

View or enter a description of the custom rule.

MDX

View or enter the MDX expression that defines the custom rule. You can also use the edit button (**...**) to open the MDX builder, where you can create an MDX expression.

You can allow access to cells for a dimension member by including the following elements in the expression:

- Dimension name
- MDX **CurrentMember** function
- MDX **Name** function
- Equality operator (=)
- Member name (If you specify a member name that is not unique within the dimension, use the **UniqueName** function instead of the **Name** function.)

    Cells for other members in the dimension are not accessible.

    For example, to allow access to cells for the USA member in the Geography dimension but deny access to cells for all other members in that dimension, use the following expression:

  ```
  Geography.CurrentMember.Name = "USA"
  ```

    Alternatively, you can deny access to cells for a member by substituting the inequality operator (<>). Cells for other members in the dimension are accessible. For example, to deny access to cells for the Brazil member in the Geography dimension but allow access to cells for all other members in that dimension, use the following expression:

  ```
  Geography.CurrentMember.Name <> "Brazil"
  ```

    To include a member's descendants in the access criteria, add the **Ancestor** function and the member's level name to the expression. For example, to deny access to cells for the Brazil member and all its descendants in the Geography dimension but allow access to cells for all other members in that dimension, use the following expression:

```
Ancestor(Geography.CurrentMember,[Country]).Name <> "Br
```

To include multiple dimensions from the cube in the access criteria, write an expression for each dimension and combine them into one expression using AND or OR. (Dimensions that are excluded from the expression do not impose restrictions on cell access.) For example, the following expression allows access to cells for the Tokyo member and its descendants in the Geography dimension but denies access to cells for all other members in that dimension. It also denies access to cells for the Sales measure, including Sales cells for Tokyo and its descendants.

```
Ancestor(Geography.CurrentMember,[City]).Name = "Tokyo"
  Measures.CurrentMember.Name <> "Sales"
```

For more information and examples, see Custom Rules in Cell Security.

Check

Checks the cell security syntax.

OK

Closes the dialog box and displays the **Cube Role** dialog box. To save the custom rule, click **OK** in the **Cube Role** dialog box.

See Also

MDX Function List

Cube Editor - Data View

Use this tool to browse a cube's data and examine and edit a cube's structure. You can also use Cube Editor and connected dialog boxes to perform various procedures.

Cube Editor appears in two views, data and schema. Both views include the tree pane and properties pane. The data view includes the **Data** tab, and the schema view includes the **Schema** tab. You can switch from one view to another by clicking the **Data** tab or the **Schema** tab at the bottom of Cube Editor. You can also click **Data** or **Schema** on the **View** menu.

This topic describes the data view. For information about the schema view, see Cube Editor - Schema View. The data view is shown here.

Cube Editor – Data View appears when you do either of the following:

- In the Analysis Manager tree pane, right-click a cube, and then click **Edit**.

- In the Analysis Manager tree pane, click a cube, and then on the **Action** menu, click **Edit**.

- Cube Editor has five areas. For more details about the areas, click a link in the following table.

To do this	See
Perform commands available in Cube Editor menus	Menus (Cube Editor Data View)
Perform common actions represented by icons on the Cube Editor toolbar	Toolbar (Cube Editor Data View)
View cube objects displayed in the tree pane	Tree Pane (Cube Editor Data View)
View and modify properties of	Properties Pane (Cube Editor

the object selected in the Cube Editor tree pane View descriptions of properties and the objects to which they apply	Data View)
View the cube data in a table format. Exchange or move dimensions Slice through a dimension, or drill down into a member	Data Tab (Cube Editor Data View)

See Also

Building a Cube with the Editor

Creating and Maintaining Private Dimensions

Creating Calculated Members

Specifying Drillthrough Options

Menus (Cube Editor Data View)

The following options are available through menus in Cube Editor.

Menu	Option	Description
File	**New Cube - Wizard**	Starts the Cube Wizard so you can create a new cube.
	New Cube - Editor	Displays the **Choose a Fact Table** dialog box so you can begin building a cube with the editor.
	Save	Saves the cube.If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
	Save As	Saves the cube under a different name.
	Exit	Closes Cube Editor.
Edit	**Rename**	Renames the selected object.
	Delete	Deletes the selected object.
	Remove Join	Removes all joins from the selected column (schema view only).
View	**Schema**	Displays the schema view, including the **Schema** tab.
	Data	Displays the data view, including the **Data** tab.
	Properties	Expands or collapses the properties pane.
Insert	**Tables**	Displays the **Select Table** dialog box so you can add tables to the cube.
	Dimension - New	Starts the Dimension Wizard so you can create a new dimension.
	Dimension - Existing	Displays Dimension Manager so you can add existing shared dimensions to the cube.
	Level	Displays the **Insert Level** dialog box so you can add a new level.

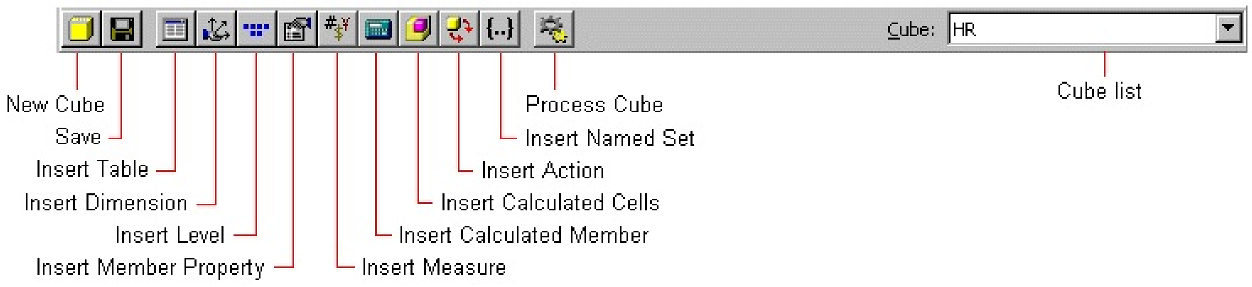
	Measure	Displays the **Insert Measure** dialog box so you can add a new measure.
	Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property.
	Calculated Member	Displays Calculated Member Builder so you can begin creating calculated members.
	Action	Starts the Action Wizard so you can create a new action.
	Named Set	Displays Named Set Builder so you can define a new named set.
	Join	Adds a join to the selected column (schema view only).
Tools	**Process Cube***	Processes the cube.
	Design Storage	Starts the Storage Design Wizard so you can select storage options and design aggregations for the cube.
	Drillthrough Options*	Displays the **Drillthrough Options** dialog box so you can select the columns that are displayed when a drillthrough is executed.
	Count Dimension Members	Provides an estimate of the number of members in a dimension. This estimate is used by the aggregation design algorithm when it determines what aggregations to create for this dimension.
	Validate Cube Structure	Verifies that the cube structure does not contain invalid components that would prevent processing of the cube.
	Optimize Schema	Simplifies a cube's schema for maximum performance during processing by removing unnecessary joins between dimension and fact tables and eliminating the need to read dimension tables during processing. Several conditions must be met before this option can be used. For more information,

		see Optimizing Cube Schemas. To remove other tabular joins, use the **Remove Join** command.
Help	**Help on Cube Editor**	Displays a Help topic about Cube Editor.
	Contents and Index	Opens SQL Server Books Online.

* If this option is not available, the cube structure is not valid. You can try to determine the problem by clicking **Validate Cube Structure** on the **Tools** menu.

Toolbar (Cube Editor Data View)

Use the Cube Editor toolbar buttons to perform common operations.



Button	Description
New Cube	Starts the Cube Wizard so you can create a new cube.
Save	Saves the cube.If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
Insert Table	Displays the **Select table** dialog box so you can add tables to the cube.
Insert Dimension	Displays Dimension Manager so you can add existing dimensions to the cube. In Dimension Manager you can start the Dimension Wizard to create a new dimension.
Insert Level	Displays the **Insert Level** dialog box so you can add a new level to a private dimension. Use Dimension Editor to add a new level to a shared dimension.
Insert Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property in the selected level of a private dimension. Use Dimension Editor to add a new member property in a level of a shared dimension.
Insert Measure	Displays the **Insert Measure** dialog box so you

	can add a new measure.
Insert Calculated Member	Displays Calculated Member Builder so you can begin creating calculated members.
Insert Calculated Cells	Starts the Calculated Cells Wizard so you can define calculated cells. The wizard adds the new calculated cells definition to the Calculated Cells folder.
Insert Action	Starts the Action Wizard so you can create a new action.
Insert Named Set	Displays Named Set Builder so you can define a new named set.
Process Cube	Displays the **Process a Cube** dialog box so you can select cube processing options.

Tree Pane (Cube Editor Data View)

Use the tree pane to display objects in a cube. Right-click an object to see a shortcut menu for that object.

The following table lists how to access available information about cube objects.

Right-click	Shortcut menu option	Description
Cube	**New Cube**	Displays the **Choose a Fact Table** dialog box so you can begin building a cube with the editor.
	Process Cube	Processes the cube.
	New Dimension	Starts the Dimension Wizard so you can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
	New Measure	Displays the **Insert Measure** dialog box so you can add a new measure.
Dimensions folder	**New Dimension**	Starts the Dimension Wizard so you can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
Dimension	**New Dimension**	Starts the Dimension Wizard so you can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
	Remove	Removes the dimension from the cube.
	Browse	Displays the dimension members in Dimension Browser.
Level	**New Dimension**	Starts the Dimension Wizard so you

		can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
Measures folder	**New Measure**	Displays the **Insert Measure** dialog box so you can add a new measure.
Measure	**New Measure**	Displays the **Insert Measure** dialog box so you can add a new measure.
	Delete	Deletes the measure.
	Rename	Renames the measure.
Calculated members folder	**New Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
Calculated member	**New Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
	Edit	Displays the calculated member in Calculated Member Builder so you can edit it.
	Delete	Deletes the calculated member.
	Rename	Renames the calculated member.
Calculated Cells folder	**New Calculated Cells**	Displays the Calculated Cells Wizard so you can begin creating calculated cells.
	Import Calculated Cells	Displays the **Import Calculated Cells dialog box so you can import calculated** cells from a source cube or another cube.
Calculated cells	**New Calculated Cells**	Displays the Calculated Cells wizard so you can begin creating calculated cells.
	Edit	Displays the Calculated Cells Wizard so you can edit the selected calculated cells definition.
	Rename	Renames the selected calculated cells definition.

	Delete	Deletes the selected calculated cells action.
Actions folder*	**New Action**	Starts the Action Wizard so you can create a new action.
Action	**New Action**	Starts the Action Wizard so you can create a new action.
	Edit	Starts the Action Wizard so you can edit it.
	Delete	Deletes the action.
	Rename	Renames the action.
Named Sets folder	**New Named Set**	Displays Named Set Builder so you can create new named sets.
Named set	**New Named Set**	Displays Named Set Builder so you can create new named sets.
	Edit	Starts Named Set Builder so you can edit the set.
	Rename	Renames the named set.

* A limit of 32 actions can be displayed when the Actions folder is expanded.

Properties Pane (Cube Editor Data View)

Use the properties pane to display the properties of the object selected in the tree pane. To display the properties pane, click **Properties** beneath the tree pane.

Each type of object contains a different set of properties. Use the properties pane to modify the property settings for the selected object. For shared dimensions, their levels, and their member properties, some properties are read-only and must be changed in Dimension Editor.

The following table describes the properties displayed in the properties pane.

Object	Property	Description
Action	**Action Type**	Indicates the particular type of action. For descriptions of the different types of actions, see CREATE ACTION Statement.
	Application	Used to provide additional information to the application that performs the action.
	Caption Expression	A Multidimensional Expressions (MDX) expression used by the client application as a caption for the action.
	Description	Used to provide a description for the action.
	Invocation	Used to determine how the action will be activated. • **Interactive**: Actions run interactively when the end user activates an associated element in the user interface. This is the default setting and the only type supported by Cube Browser in Microsoft® SQL Server™ 2000 Analysis Services. • **On Open**: Actions run automatically when the client opens a cube.

		• **Batch**: Actions are run by the client in a batch job. These actions provide a way to check for exceptions.
	Name	The name of the action. This is displayed when selecting the target object for the action.
	Value	The MDX expression that defines the action.
Calculated cells	**BackColor**	Indicates the background color of the cells defined by the cell coordinates and condition.
	Calculation Condition	Contains an MDX conditional expression that defines the calculation subcube cells that are computed with the calculation formula by testing each cell.
	Calculation Subcube	Contains an MDX set expression that defines the subset of the cube data that is computed using the calculation formula.
	Calculation Value	Contains an MDX expression that provides the value for each cell in the calculation subcube.
	Description	Contains the text string that describes the calculated cells definition.
	Disabled	Determines whether the cell calculation is disabled. The default value is **False**.
	FontName	Indicates the font name of the cells defined by the cell coordinates and condition.
	FontSize	Indicates the font size of the cells defined by the cell coordinates and condition.
	FontFlags	Indicates the font flags of the cells defined by the cell coordinates and condition.
	Calculation Pass Number	Indicates the calculation pass in which this calculation is executed.
	Calculation Pass Depth	Determines how many calculation passes are required to fully compute a calculated cells definition.
	ForeColor	Indicates the foreground color of the cells

		defined by the cell coordinates and condition.
	Format String	Contains the format string of the cells defined by the cell coordinates and condition.
	Name	Contains the name of the calculated cells definition. This property is read-only.
	Solve Order	Contains a number representing the order of evaluation of the calculated cells.
	Visible	Determines whether the calculated cells definition is visible in the schema rowset. The default value is **True**.
Calculated member	**BackColor**	The background color of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	FontFlags	The font flags of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	FontName	The font of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	FontSize	The font size of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support

		this property.
	ForeColor	The foreground color of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	Format String	The format for displaying cell values. A list of common formats is displayed when you click **Format String** and display the dropdown list. This property accepts the same values as the **Display Format** property of measures. For more information, see Display Formats.
	Name	The name of calculated member. The name is displayed when end users browse the cube.
	Non Empty Behavior	Stores the name of the measure used to resolve NON EMPTY queries in MDX. If the **Non Empty Behavior** property is blank, the calculated member must be evaluated repeatedly to determine if a member is empty. If the **Non Empty Behavior** property contains the name of a measure, the calculated member is treated as empty if the specified measure is empty.
	Parent Dimension	The dimension that includes the calculated member. If the calculated member is a measure, specify **Measures**.
	Parent Member	The member that includes the calculated member. The parent member determines the location of the calculated member in the dimension structure. This property must be null if the parent dimension is **Measures** or a one-level dimension.
	Solve Order	The order in which the calculated member is resolved in case of intersection with other

		calculated members. Valid values are 0 (zero) and positive or negative integers. Calculated members with lower **Solve Order** values have precedence in resolution.
	Value	An MDX expression that defines the values of the calculated member. The values are displayed when end users browse the cube.
	Visible	Determines whether the cube is visible in a list of cubes.
Cube	**Aggregation Prefix**	The prefix appended to aggregation names for the cube's partitions, provided that the partition's aggregation prefix begins with a plus sign (+). In this case, this property's value is appended to the beginning of the partition's aggregation prefix. If the partition's aggregation prefix does not begin with a plus sign, this property is ignored. To access the aggregation prefix for a partition, in the Analysis Manager tree pane, right-click the partition, click **Edit**, advance to the **Finish** step of the Partition Wizard, and then click **Advanced**.
	Data Source	The data source for the cube. The partitions of the cube can have different data sources.
	Default Measure	The measure that is returned by queries when no measure is displayed on an axis and no slicing measure is specified. If no default measure is specified, an arbitrary measure is the default measure.
	Description	The description of the cube.
	Fact Table	The fact table for the cube. The partitions of the cube can have different fact tables.
	Fact Table Size	The number of rows in the fact table of the cube at the time they were last counted by Analysis Services, or a user-provided estimate of the number of rows.
	Key Error	Limit for the number of dimension key errors.

	Limit	The default is 0. Cube processing is halted and cancelled when the limit is exceeded, provided that the **Stop Processing on Key Errors** property of the cube is **Yes**. If you select **Yes** and a **Key Error Limit** value greater than 0, and processing completes, the data in the cube does not reflect the entire fact table. The **Key Error Limit** property is ignored if the **Stop Processing on Key Errors** property is **No**.
	Key Error Log File	Path and file name of the log file for dimension key errors.
	Name	The name of the cube.
	Processing Optimization Mode	Values of the **Processing Optimization Mode** property are **Regular** (processed data is available after all aggregations have been computed) or **Lazy Aggregations** (processed data is available immediately after data has been loaded). This property only applies to MOLAP partitions of a cube.
	Source Table Filter	The WHERE clause expression applied to the partitions' fact tables to limit the data in the cube. This property provides defaults for the filters in the partitions of the cube. These filters override this property. To access a filter in a partition, in the Analysis Manager tree pane, right-click the partition, click **Edit**, advance to the **Finish** step of the Partition Wizard, and then click **Advanced**.
	Stop Processing on Key Errors	If you select **Yes**, processing is halted and cancelled when the limit for the number of dimension key errors is exceeded. This limit is specified in the **Key Error Limit** property of the cube. A dimension key error occurs when a fact table row is encountered that contains a foreign key value not present in the joined primary key column of a dimension table.

		If you select **No**, dimension key errors never halt or cancel cube processing regardless of the number of errors encountered. If one or more dimension key errors are encountered, the data in the cube does not reflect the entire fact table.
	Visible	Indicates whether the cube is visible when end users browse the list of available cubes.
Dimension	**Aggregation Usage**	Indicates the levels for which aggregation data is calculated. Choices in the list are: <ul><li>Standard: The levels for which aggregation data is calculated depend on the variety of the dimension. For regular dimensions, aggregation data is calculated for all levels unless the **Changing** property of the dimension is **True**, in which case aggregation data is not calculated for levels between the top and bottom levels. For parent-child dimensions, aggregation data is calculated for the bottom level displayed to end users and the (All) level, if any. For virtual dimensions, aggregation data is not calculated for any level.</li><li>Top Level Only: Aggregation data is calculated for only the top level. This value is not valid for parent-child dimensions or dimensions with their **Changing** property set to **True** unless the **All Level** property of the dimension is set to **Yes**.</li><li>Bottom Level Only: Aggregation data</li></ul>

		is calculated for only the bottom level. • Top and Bottom Levels: Aggregation data is calculated for only the top and bottom levels. This value is not valid for parent-child dimensions or dimensions with their **Changing** property set to **True** unless the **All Level** property of the dimension is set to **Yes**. • Custom: Aggregation data is calculated for only the levels with an **Enable Aggregations** property of **Yes**. Custom is not valid for parent-child dimensions or dimensions with their **Changing** property set to **True**.
	All Caption	The name of the member in the (All) level.
	All Level	Indicates whether the dimension contains an (All) level. If the value is **Yes**, the (All) level is the top level of the dimension but is not displayed in the Cube Editor tree pane. The (All) level contains a single member whose cell value is the aggregate of cell values for all members in the next lower level.
	All Member Formula	The custom rollup formula for the (All) level. This formula is an MDX expression that determines the cell values for the All member and overrides the **Aggregate Function** properties of measures. For more information, see Custom Rollup Formulas and Custom Member Formulas.
	Allow Duplicate Names	Indicates whether the members under a common parent can have the same name.

	Changing	Indicates whether the dimension is optimized for frequent changes. If the value is set to **True**, query performance may be slower. However, levels and members below the top level and above the bottom level can be added, moved, and deleted. This eliminates or reduces the subsequent processing requirement, and also minimizes interruptions of end users' access to the cubes that include that dimension. Several conditions must exist before the **Changing** property can be set to **True**. For more information, see Changing Dimensions.
	Data Member Caption Template	Controls the names of data members when the **Members With Data** property of the dimension is set to **Nonleaf data visible**. When you type a value that includes an asterisk (*), the name of each data member will be the value with the asterisk replaced by the name of the parent member. The **Data Member Caption Template** property is available only for parent-child dimensions.
	Data source	The data source that contains the dimension table(s).
	Default Member	The member that slices the datasets returned by queries when the dimension is not displayed on an axis and no slicing member in the dimension is specified. If no default member is specified, and the **All Level** property of the dimension is **Yes**, the member indicated by the **All Caption** property is the default member. If no default member is specified, and the **All Level** property of the dimension is **No**, an arbitrary member of the highest level is the default member.
	Depends on Dimension	For virtual dimensions, the dimension that supplies the member properties or columns on

		which the levels of the virtual dimension are based. For dimensions that are not virtual, the dimension according to which aggregation design is optimized. A dimension in the value of this property is advantageous when the cross product of the two dimensions' members results in a significant percentage of combinations that cannot coexist. For example, the **Depends on Dimension** property of a Customer Gender dimension is Customers. Fifty percent of the combinations resulting from the cross product of the dimensions' lowest-level members cannot coexist because a customer can have only one gender. For more information, see Dependent Dimensions.
	Description	Provides a description of the dimension.
	Enable All Level Aggregations	Determines whether to consider the (All) level when designing aggregations to optimize performance.
	Enable Real-Time Updates	Indicates whether the dimension supports real-time updates to underlying dimension tables.
	Member Keys Unique	Indicates whether member keys are unique within the dimension. If this property is set to **True**, the **Member Keys Unique** property for every level in the dimension is set to **True**. If the value of this property is changed, you must reprocess the current cube using the **Full Process** option. For more information, see Processing Cubes.
	Member Names Unique	Indicates whether member names are unique within the dimension. If the value is **True**, internal member names omit qualifying level names and member names. These internal

		member names are used in MDX expressions. If this property is set to **True**, the **Member Names Unique** property for every level in the dimension is set to **True**.
	Members With Data	Indicates whether nonleaf members are allowed to have associated fact table data. If they are allowed, this property also indicates whether children of nonleaf members are created to display this data. Valid values are: <ul><li>Leaf members only: Only leaf members can have associated fact table data. If a nonleaf member has associated fact table data, processing fails. This value is the default.</li><li>Nonleaf data hidden: Nonleaf members can have associated fact table data. This data is not represented among the nonleaf members. Consequently, it might appear to end users that values aggregate incorrectly.</li><li>Nonleaf data visible: Nonleaf members can have associated fact table data. This data is represented among the descendants of the nonleaf members by the creation of a child for each nonleaf member. This child, called a data member, is a leaf member and has a value equal to the aggregate of its parent's associated fact table data. The data members' names are controlled by the **Data Member Caption Template** property.</li></ul>This property is available only for parent-child

		dimensions.
	Name	The name of the dimension.
	Source Table Filter	A WHERE clause expression applied to the dimension table to limit the members in the dimension. For example, in the Store dimension supplied with Analysis Services, to include only the Canada member and its descendants, type: "store"."store_country" = 'Canada'
	Storage Mode	The type of storage for the dimension. If the value is **MOLAP** (multidimensional OLAP), the dimension data is stored in a multidimensional structure on the Analysis server. If the value is **ROLAP** (relational OLAP), the dimension data is the dimension table itself. **MOLAP** provides better performance and is recommended except for extremely large dimensions, that is, dimensions that have roughly 5 to 10 million members. In order to select **ROLAP**, the **Member Keys Unique** property of the lowest level must be **True**. Restrictions also apply to the dimension's **Aggregation Usage** property in all cubes in which the dimension is used. For more information, see Dimension Storage Modes.
	Type	The type of the dimension. Standard is the default. This property indicates to client applications the kind of information in the dimension.
	Virtual	Indicates whether the dimension is a virtual dimension, that is, one based on the tables and columns of another dimension). If this property is set to **True**, an (All) level is automatically created for the dimension.

	Visible	Indicates whether the dimension is visible when end users browse the cube.
	Write-enabled	Indicates whether the members of the dimension can be updated while administrators browse the dimension and while end users browse the cube. The only end users that can update a write-enabled dimension are those in cube roles granted read/write access to that dimension. Only parent-child dimensions can be write-enabled. For more information, see Write-Enabled Dimensions.
Level	**Custom Members**	Indicates whether custom member formulas are used to determine cell values of the members. For more information, see Custom Rollup Formulas and Custom Member Formulas. To view the custom member formulas, right-click the dimension, click **Browse,** and then see the custom member formula pane.
	Custom Member Options	Indicates whether calculation options can be defined for custom members on this level and unary operators on the following level. Calculation options are stored in a column in the dimension table. To set this property to **True**, you must first set the **Custom Members** property to **True**.
	Custom Rollup Formula	An MDX expression that determines the members' cell values and overrides the **Aggregate Function** properties of measures. For more information, see Custom Rollup Formulas and Custom Member Formulas. The structure of a cube will not be valid if the cube has a measure with the **Distinct Count** aggregate function, or if you add a custom rollup formula to a level of the cube.
	Description	Provides a description of the level.

	Disabled	Indicates whether the level is included in the cube. This property is not available for parent-child dimensions. A level whose **Disabled** property is **Yes** cannot be referenced explicitly in calculated members and other MDX expressions.
	Enable Aggregations	Indicates whether aggregations are calculated for the level. The value of this property can be changed only if the **Aggregation Usage** property of the dimension is **Custom**. The value of this property is always **No** for levels in virtual dimensions. This property is not available for parent-child dimensions.
	Grouping	Indicates whether the level contains member groups. Member groups can be used to satisfy the maximum limit of 64,000 members under a parent. To group members, create a new level immediately above and identical to the level that exceeds the limit, and then set the **Grouping** property of the new level to **Automatic**. For more information, see Member Groups. If the level is in a shared dimension, you must group members in Dimension Editor.
	Hide Member If	Determines which members are hidden from end users browsing the cube. Hidden members support ragged dimensions, which contain logical gaps in member lineage, by hiding the members that occupy the gaps. Valid values are: • Never hidden: No members are hidden. • No name: Every member whose name is null or an empty string is hidden. • Parent's name: Every member with the same name as its parent is hidden.

		<ul><li>Only child with no name: Every member that is an only child and whose name is null or an empty string is hidden.</li><li>Only child with parent's name: Every member that is an only child and has the same name as its parent is hidden.</li></ul>This property is not available for parent-child dimensions.
	Key Data Size	The size (in bytes) of the columns that store member keys in aggregations. Member keys are copied from the column specified in the **Member Key Column** property.
	Key Data Type	The data type of the columns that store member keys in aggregations. Member keys are copied from the column specified in the **Member Key Column** property.
	Level Naming Template	Determines the level names displayed to end users browsing the cube. This property is available only for parent-child dimensions. If the level is in a private dimension, click this property and click the edit (**...**) button to display the **Level Naming Template** dialog box.
	Level Type	The type of the level. Regular is the default. The following values are used only in dimensions whose **Type** property is **Time**: Years, Half-Years, Quarters, Months, Weeks, Days, Hours, Minutes, Seconds, and Time-Undefined. The **Level Type** property indicates to client applications the kind of information in the level. This property is not displayed for parent-child dimensions.

	Member Count	The number of members in the level at the time members were last estimated by Analysis Services, or a user-provided estimate of the member count. If the level is in a private dimension, you can update the member count by clicking **Count Dimension Members** on the **Tools** menu.
	Member Key Column	The column that contains the member keys. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Keys Unique	Indicates whether member keys are unique within the level. This property is not available for parent-child dimensions. If the value of this property is changed, you must reprocess the current cube using the **Full Process** option. For more information, see Processing Cubes.
	Member Name Column	The column that contains the member names, which are displayed to end users browsing the cube. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Names Unique	Indicates whether member names are unique within the level. If the value is **True**, internal member names omit qualifying member names. These internal member names are used in MDX

		expressions.
	Name	The name of the level.
	Order By	The sort order for displayed members. You can sort by member name, member key, or any member property defined for the level.
	Parent Key Column	The column that contains the keys for the members' parents. This property is available only for parent-child dimensions.
	Root Member If	Indicates the criteria by which members of the highest level (ignoring the (All) level, if any) are identified. The following values are valid: • Parent is blank, self, or missing: A member is in the highest level if any of the following three criteria is met. • Parent is blank: A member is in the highest level if its value in the **Parent Key Column** is null or 0 (zero). • Parent is self: A member is in the highest level if its value in the **Parent Key Column** is equal to its value in the **Member Key Column**. • Parent is missing: A member is in the highest level if its value in the **Parent Key Column** does not exist in the **Member Key Column**. This property is available for parent-child dimensions only.
	Skipped Levels	Indicates the column that contains the number of levels between a member and its parent,

	Column	excluding the member and parent. Valid values in the column are 0 (zero) and positive integers. This property is available only for parent-child dimensions.
	Unary Operators	Enables unary operators or custom rollup operators that control how level members are aggregated into the value of their parent member. For more information, see Custom Rollup Operators. To view the custom member formulas, on the **View** menu click **Data**, and then see the value beside UNARY_COLUMN in the Member properties pane. The structure of the cube will not be valid if a cube has a measure with the **Distinct Count** aggregate function, or if you add custom rollup operators to a level of the cube.
	Visible	Indicates whether the level is visible to end users browsing the cube. To set this property to **False**, you must first set the **Member Keys Unique** property of all lower levels to **True**. The **Visible** property is not available for parent-child dimensions. Unlike other objects whose **Visible** property is **False**, a level cannot be explicitly referenced in calculated members and other MDX expressions.
Measure	**Aggregate Function**	The function used to aggregate measure values. Valid values are: • **Sum**: The measure value for a dimension member is calculated by adding the values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants.

		• **Count**: The measure value for a dimension member is calculated by adding the number of values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. Null values are not counted. • **Min**: The measure value for a dimension member is the lowest of the values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. • **Max**: The measure value for a dimension member is the highest of the values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. • **Distinct Count**: The measure value for a dimension member is calculated by adding the number of different values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. A cube can contain only one measure with the **Distinct Count** aggregate function.

		If a cube has a measure with the **Distinct Count** aggregate function, the structure of the cube will not be valid if you add a custom rollup operator or expression to a level of the cube. Measure values for the intersections of members from different dimensions are also calculated using the selected aggregate function. For more information about aggregate functions, see Aggregate Functions. A multiple-partition cube can have multiple fact tables, so a measure can have multiple source columns. (Note that each source column must have the same column name.) In this case the aggregate function spans all the source columns.
	Data Type	The data type of the columns that store measure values in aggregations.
	Description	The description of the measure.
	Display Format	The format of the measure values displayed to end users browsing the cube. For more information, see Display Formats.
	Name	The name of the measure. The name is displayed to end users browsing the cube.
	Source Column	In the fact table of the default partition of the cube, the column that stores the measure values before they are aggregated.
	Visible	Indicates whether the measure is visible to end users browsing the cube.
Member property	**Caption**	The caption used to display the member property.
	Data Size	The maximum number of characters allowed in the column that stores the member property values. Allows dimension tables to be linked to

		fact tables when the data types of the columns do not match but the contents represent the same values.
	Data Type	The data type of the column that stores the member property values. Allows dimension tables to be linked to fact tables when the data types of the columns do not match but the contents represent the same values.
	Description	The description of the member property.
	Language	The language used to display the member property.
	Name	The name of the member property.
	Source Column	The column that stores the values of the member property. This column must be in one of the dimension tables for the dimension that contains the member property.
	Type	Indicates to client applications the type of information in the member property values.
	Visible	Indicates whether the member property is visible to end users browsing the cube.
Named set	**Name**	The name of the named set.
	Value	The MDX expression that defines the named set.

Data Tab (Cube Editor Data View)

Use the **Data** tab to display the cube data. The default view in the data viewing pane shows the data in table format with one dimension across the column headings and another dimension down the left column. (Measures are treated as a single dimension for this purpose.) The remaining dimensions of the cube are displayed in the data slicing pane.

The white cells in the data viewing pane represent the measure values according to the members that appear in the members boxes for all the dimensions shown in the data slicing pane. In the preceding example, the measure values reflect the All member of every dimension in the data slicing pane except Time. The members box for the Time dimension displays 1997, which indicates that the displayed measure values are limited to 1997. Later in this topic, an example shows you how to slice through dimensions in the data slicing pane to limit the measure values that are displayed.

The data in the **Data** tab is either internally generated sample data or actual data from the cube.

Sample data appears when the cube has been changed since it was last processed, including changes made during your current editing session. You can browse the sample data to check the design of your cube as you are building or editing it. You can see how the data is organized according to the current structure of the cube.

Actual data appears when the cube structure has not been changed since it was last processed. To browse the actual data, you can drill down or slice through your data.

Working in the Data Tab

To browse the cube data in the data viewing pane by any combination of dimensions, drag measures or dimensions from the data slicing pane onto either axis in the data viewing pane.

To replace a dimension in the data viewing pane with another dimension

- Drag the dimension from the data slicing pane to the dimension in the data viewing pane that you want to replace.

The pointer is in the correct position when it appears as follows.

Exchange a slicer dimension with a row dimension.

-or-

Exchange a slicer dimension with a column dimension.

The dimension in the data viewing pane moves up to the data slicing pane and the new one takes its place.

To move a dimension to the data viewing pane

- Drag the dimension from the data slicing pane to a cell below (or beside) the existing dimension.

The pointer is in the correct position when it appears as follows.

Move a slicing dimension to a row.

-or-

Move a slicing dimension to a column.

The new dimension appears under (or beside) the existing dimension, expanding your data viewing pane.

To change the order of dimensions in the data viewing pane

- Drag the dimension and drop it on the dimension where you want it to be located.

To remove a dimension from the data viewing pane

- Drag the dimension you want to remove from the data viewing pane to the data slicing pane.

To drill down into a member

- Double-click a dimension member in the data viewing pane. If the member has members beneath it, they are displayed.

Only the member you double-click is expanded. The other members retain their current levels in the data viewing pane. In the following example, double-click the Drink member to drill down into its child members.

If you want to display all the members in a level, double-click the parent level name in the data viewing pane. Level names are depicted with the shading of a button (for example, Product Family in the preceding example).

To slice through a dimension

- To change the member in a dimension in the data slicing pane, click the down arrow on the members box, expand the members, and then select a new member. The cube data in the data viewing pane changes to reflect the change in the dimension member.

The **Data** tab has an internal memory limit that you may reach if you attempt to browse too much data or drill down too deeply. When you reach the limit, the following message is displayed:

Unable to display current view of cube.
Unable to Allocate Memory For Flexgrid.

The limit cannot be increased by adding or allocating more memory. If you reach the limit, reduce the amount or depth of data you are attempting to browse, or use another browser.

Cube Editor - Schema View

Use this tool to view and edit the schema of a cube and examine and edit the structure of a cube. You can also use Cube Editor and connected dialog boxes to perform various procedures.

Cube Editor appears in two views, data and schema. Both views include the tree pane and the properties pane, but the data view includes the **Data** tab, whereas the schema view includes the **Schema** tab. You can switch from one view to another by clicking the **Data** tab or the **Schema** tab at the bottom of Cube Editor. You can also click **Data** or **Schema** on the **View** menu.

This topic describes the schema view. For information about the data view, see Cube Editor - Data View. The schema view is shown here.

Cube Editor appears when you do either of the following:

- In the Analysis Manager tree pane, right-click a cube, and then click **Edit**.

- In the Analysis Manager tree pane, click a cube, and then on the **Action** menu, click **Edit**.

Cube Editor has five areas. For more information about the areas, click a link in the following table.

To do this	See
Perform commands available in the Cube Editor menus	Menus (Cube Editor Schema View)
Perform common actions represented by icons on the Cube Editor toolbar	Toolbar (Cube Editor Schema View)
View cube objects displayed in the tree pane	Tree Pane (Cube Editor Schema View)
View and modify certain properties of the object selected in the Cube	Properties Pane (Cube Editor Schema View)

Editor tree pane View descriptions of properties and the objects to which they apply	
View a the fact table and the dimension tables of a cube, and all columns in a table Add measures and shared dimensions, and create private dimensions	Schema Tab (Cube Editor Schema View)

See Also

Building a Cube with the Editor

Creating and Maintaining Private Dimensions

Specifying Drillthrough Options

Menus (Cube Editor Schema View)

The following options are available through the menus in Cube Editor.

Menu	Option	Description
File	**New Cube - Wizard**	Starts the Cube Wizard so you can create a new cube.
	New Cube - Editor	Displays the **Choose a Fact Table** dialog box so you can begin building a cube with the editor.
	Save	Saves the cube. If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
	Save As	Saves the cube under a different name.
	Exit	Closes Cube Editor.
Edit	**Edit**	Edits the selected object. This option is unavailable for objects that have no associated editor or wizard.
	Rename	Renames the selected object.
	Delete	Deletes the selected object.
	Remove Join	Removes all joins from the selected column (schema view only).
View	**Schema**	Displays the schema view, including the **Schema** tab.
	Data	Displays the data view, including the **Data** tab.
	Properties	Expands or collapses the properties pane.
Insert	**Tables**	Displays the **Select Table** dialog box so you can add tables to the cube.
	Dimension - New	Starts the Dimension Wizard so you can create a new dimension.
	Dimension -	Displays Dimension Manager so you can

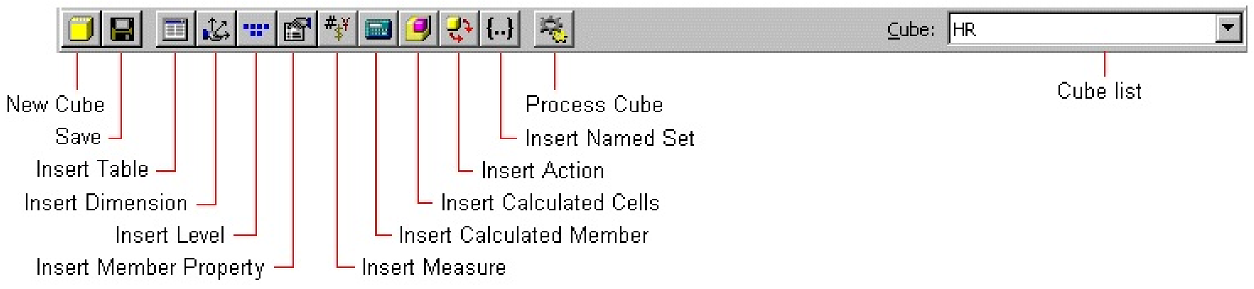
	Existing	add existing shared dimensions to the cube.
	Level	Displays the **Insert Level** dialog box so you can add a new level.
	Measure	Displays the **Insert Measure** dialog box so you can add a new measure.
	Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property.
	Calculated Member	Displays Calculated Member Builder so you can begin creating calculated members.
	Calculated Cells	Displays the Calculated Cells Wizard so you can begin creating calculated cells.
	Action	Starts the Action Wizard so you can create a new action.
	Named Set	Displays Named Set Builder so you can define a new named set.
	Join	Adds a join to the selected column (schema view only).
Tools	**Process Cube***	Processes the cube.
	Design Storage	Starts the Storage Design Wizard so you can select storage options and design aggregations for the cube.
	Drillthrough Options*	Displays the **Drillthrough Options** dialog box so you can select the columns that are displayed when a drillthrough is executed.
	Count Dimension Members	Provides an estimate of the number of members in a dimension. This estimate is used by the aggregation design algorithm when it determines what aggregations to create for this dimension.
	Validate Cube Structure	Verifies that the cube structure does not contain invalid components that would prevent processing of the cube.
	Optimize Schema	Simplifies a cube's schema for maximum performance during processing by removing

		unnecessary joins between dimension and fact tables and eliminating the need to read dimension tables during processing. Several conditions must be met before this option can be used. For more information, see Optimizing Cube Schemas. To remove other tabular joins, use the **Remove Join** command.
Help	**Help on Cube Editor**	Displays a Help topic about Cube Editor.
	Contents and Index	Opens SQL Server Books Online.

* If this option is not available, the cube structure is not valid. You can try to determine the problem by clicking **Validate Cube Structure** on the **Tools** menu.

Toolbar (Cube Editor Schema View)

Use the toolbar buttons to perform common operations.



Button	Description
New Cube	Starts the Cube Wizard so you can create a new cube.
Save	Saves the cube.If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
Insert Table	Displays the **Select table** dialog box so you can add tables to the cube.
Insert Dimension	Displays Dimension Manager so you can add existing dimensions to the cube. In Dimension Manager you can start the Dimension Wizard to create a new dimension.
Insert Level	Displays the **Insert Level** dialog box so you can add a new level to a private dimension. Use Dimension Editor to add a new level to a shared dimension.
Insert Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property in the selected level of a private dimension. Use Dimension Editor to add a new member property in a level of a shared dimension.
Insert Measure	Displays the **Insert Measure** dialog box so you can add a new measure.
Insert Calculated Member	Displays Calculated Member Builder so you can begin creating calculated members.

Insert Calculated Cells	Starts the Calculated Cells Wizard so you can define calculated cells. The wizard adds the new calculated cells definition to the Calculated Cells folder.
Insert Action	Starts the Action Wizard so you can create a new action.
Insert Named Set	Displays Named Set Builder so you can define a new named set.
Process Cube	Displays the **Process a Cube** dialog box so you can select cube processing options.

Tree Pane (Cube Editor Schema View)

Use the tree pane to display objects in a cube. Right-click an object to see a shortcut menu for that object.

The following table lists how to access available information about cube objects.

Right-click	Shortcut menu option	Description
Cube	**New Cube**	Displays the **Choose a Fact Table** dialog box so you can begin building a cube with the editor.
	Process Cube	Processes the cube.
	New Dimension	Starts the Dimension Wizard so you can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
	New Measure	Displays the **Insert Measure** dialog box so you can add a new measure.
Dimensions folder	**New Dimension**	Starts the Dimension Wizard so you can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
Dimension	**New Dimension**	Starts the Dimension Wizard so you can create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
	Remove	Removes the dimension from the cube.
	Browse	Displays the dimension members in Dimension Browser.
Level	**New Dimension**	Starts the Dimension Wizard so you can

		create a new dimension.
	Existing Dimensions	Displays Dimension Manager so you can add existing shared dimensions to the cube.
Measures folder	**New Measure**	Displays the **Insert Measure** dialog box so you can add a new measure.
Measure	**New Measure**	Displays the **Insert Measure** dialog box so you can add a new measure.
	Delete	Deletes the measure.
	Rename	Renames the measure.
Calculated members folder	**New Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
Calculated member	**New Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
	Edit	Displays the calculated member in Calculated Member Builder so you can edit it.
	Delete	Deletes the calculated member.
	Rename	Renames the calculated member.
Calculated Cells folder	**New Calculated Cells**	Displays the Calculated Cells Wizard so you can begin creating calculated cells.
	Import Calculated Cells	Displays the **Import Calculated Cells dialog box so you can import calculated** cells from a source cube or another cube.
Calculated cells	**New Calculated Cells**	Displays the Calculated Cells wizard so you can begin creating calculated cells.
	Edit	Displays the Calculated Cells Wizard so you can edit the selected calculated cells definition.
	Rename	Renames the selected calculated cells definition.
	Delete	Deletes the selected calculated cells

		action.
Actions folder*	**New Action**	Starts the Action Wizard so you can create a new action.
Action	**New Action**	Starts the Action Wizard so you can create a new action.
	Edit	Starts the Action Wizard so you can edit it.
	Delete	Deletes the action.
	Rename	Renames the action.
Named set	**New Named Set**	Displays Named Set Builder so you can create new named sets.
	Edit	Starts Named Set Builder so you can edit the set.
	Rename	Renames the named set.

* A limit of 32 actions can be displayed when the Actions folder is expanded.

Properties Pane (Cube Editor Schema View)

Use the properties pane to display the properties of the object selected in the tree pane. To display the properties pane, click **Properties** beneath the tree pane.

Each type of object contains a different set of properties. Use the properties pane to modify the property settings for the selected object. For shared dimensions, their levels, and their member properties, some properties are read-only and must be changed in Dimension Editor.

The following table describes the properties displayed in the properties pane.

Object	Property	Description
Action	**Action Type**	Indicates the particular type of action. For descriptions of the different types of actions, see CREATE ACTION Statement.
	Application	Used to provide additional information to the application that performs the action.
	Caption Expression	A Multidimensional Expressions (MDX) expression used by the client application as a caption for the action.
	Description	Used to provide a description for the action.
	Invocation	Used to determine how the action will be activated. • **Interactive**: Actions run interactively when the end user activates an associated element in the user interface. This is the default setting and the only type supported by Cube Browser in Microsoft® SQL Server™ 2000 Analysis Services. • **On Open**: Actions run automatically when the client opens a cube.

		• **Batch**: Actions are run by the client in a batch job. These actions provide a way to check for exceptions.
	Name	The name of the action. This is displayed when selecting the target object for the action.
	Value	The MDX expression that defines the action.
Calculated cells	**BackColor**	Indicates the background color of the cells defined by the cell coordinates and condition.
	Calculation Condition	Contains an MDX conditional expression that defines the calculation subcube cells that are computed with the calculation formula by testing each cell.
	Calculation Subcube	Contains an MDX set expression that defines the subset of the cube data that is computed using the calculation formula.
	Calculation Value	Contains an MDX expression that provides the value for each cell in the calculation subcube.
	Description	Contains the text string that describes the calculated cells definition.
	Disabled	Determines whether the cell calculation is disabled. The default value is **False**.
	FontName	Indicates the font name of the cells defined by the cell coordinates and condition.
	FontSize	Indicates the font size of the cells defined by the cell coordinates and condition.
	FontFlags	Indicates the font flags of the cells defined by the cell coordinates and condition.
	Calculation Pass Number	Indicates the calculation pass in which this calculation is executed.
	Calculation Pass Depth	Determines how many calculation passes are required to fully compute a calculated cells definition.
	ForeColor	Indicates the foreground color of the cells

		defined by the cell coordinates and condition.
	Format String	Contains the format string of the cells defined by the cell coordinates and condition.
	Name	Contains the name of the calculated cells definition. This property is read-only.
	Solve Order	Contains a number representing the order of evaluation of the calculated cells.
	Visible	Determines whether the calculated cells definition is visible in the schema rowset. The default value is **True**.
Calculated member	**BackColor**	The background color of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	FontFlags	The font flags of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	FontName	The font of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	FontSize	The font size of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do

		not support this property.
	ForeColor	The foreground color of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Cube Browser, the Cube Editor **Data** tab, and the Virtual Cube Editor data pane do not support this property.
	Format String	The format for displaying cell values. A list of common formats is displayed when you click **Format String** and display the dropdown list. This property accepts the same values as the **Display Format** property of measures. For more information, see Display Formats.
	Name	The name of calculated member. The name is displayed when end users browse the cube.
	Non Empty Behavior	Stores the name of the measure used to resolve NON EMPTY queries in MDX. If the **Non Empty Behavior** property is blank, the calculated member must be evaluated repeatedly to determine if a member is empty. If the **Non Empty Behavior** property contains the name of a measure, the calculated member is treated as empty if the specified measure is empty.
	Parent Dimension	The dimension that includes the calculated member. If the calculated member is a measure, specify **Measures**.
	Parent Member	The member that includes the calculated member. The parent member determines the location of the calculated member in the dimension structure. This property must be null if the parent dimension is **Measures** or a one-level dimension.
	Solve Order	The order in which the calculated member is resolved in case of intersection with other

		calculated members. Valid values are 0 (zero) and positive or negative integers. Calculated members with lower **Solve Order** values have precedence in resolution.
	Value	An MDX expression that defines the values of the calculated member. The values are displayed when end users browse the cube.
	Visible	Indicates whether the dimension is visible when end users browse the cube.
Cube	**Aggregation Prefix**	The prefix appended to aggregation names for the cube's partitions, provided that the partition's aggregation prefix begins with a plus sign (+). In this case, this property's value is appended to the beginning of the partition's aggregation prefix. If the partition's aggregation prefix does not begin with a plus sign, this property is ignored. To access the aggregation prefix for a partition, in the Analysis Manager tree pane, right-click the partition, click **Edit**, advance to the **Finish** step of the Partition Wizard, and then click **Advanced**.
	Data Source	The data source for the cube. The partitions of the cube can have different data sources.
	Default Measure	The measure that is returned by queries when no measure is displayed on an axis and no slicing measure is specified. If no default measure is specified, an arbitrary measure is the default measure.
	Description	The description of the cube.
	Fact Table	The fact table for the cube. The partitions of the cube can have different fact tables.
	Fact Table Size	The number of rows in the fact table of the cube at the time they were last counted by Analysis Services, or a user-provided estimate of the number of rows.

	Key Error Limit	Limit for the number of dimension key errors. The default is 0. Cube processing is halted and cancelled when the limit is exceeded, provided that the **Stop Processing on Key Errors** property of the cube is **Yes**. If you select **Yes** and a **Key Error Limit** value greater than 0, and processing completes, the data in the cube does not reflect the entire fact table. The **Key Error Limit** property is ignored if the **Stop Processing on Key Errors** property is **No**.
	Key Error Log File	Path and file name of the log file for dimension key errors.
	Name	The name of the cube.
	Processing Optimization Mode	Values of the **Processing Optimization Mode** property are **Regular** (processed data is available after all aggregations have been computed) or **Lazy Aggregations** (processed data is available immediately after data has been loaded). This property only applies to MOLAP partitions of a cube.
	Source Table Filter	The WHERE clause expression applied to the partitions' fact tables to limit the data in the cube. This property provides defaults for the filters in the partitions of the cube. These filters override this property. To access a filter in a partition, in the Analysis Manager tree pane, right-click the partition, click **Edit**, advance to the **Finish** step of the Partition Wizard, and then click **Advanced**.
	Stop Processing on Key Errors	If you select **Yes**, processing is halted and cancelled when the limit for the number of dimension key errors is exceeded. This limit is specified in the **Key Error Limit** property of the cube. A dimension key error occurs when a fact table row is encountered that contains a foreign key value not present in the joined

		primary key column of a dimension table. If you select **No**, dimension key errors never halt or cancel cube processing regardless of the number of errors encountered. If one or more dimension key errors are encountered, the data in the cube does not reflect the entire fact table.
	Visible	Determines whether the cube is visible in a list of cubes.
Dimension	**Aggregation Usage**	Indicates the levels for which aggregation data is calculated. Choices in the list are: • Standard: The levels for which aggregation data is calculated depend on the variety of the dimension. For regular dimensions, aggregation data is calculated for all levels unless the **Changing** property of the dimension is **True**, in which case aggregation data is not calculated for levels between the top and bottom levels. For parent-child dimensions, aggregation data is calculated for the bottom level displayed to end users and the (All) level, if any. For virtual dimensions, aggregation data is not calculated for any level. • Top Level Only: Aggregation data is calculated for only the top level. This value is not valid for parent-child dimensions or dimensions with their **Changing** property set to **True** unless the **All Level** property of the dimension is set to **Yes**.

		<ul><li>Bottom Level Only: Aggregation data is calculated for only the bottom level.</li><li>Top and Bottom Levels: Aggregation data is calculated for only the top and bottom levels. This value is not valid for parent-child dimensions or dimensions with their **Changing** property set to **True** unless the **All Level** property of the dimension is set to **Yes**.</li><li>Custom: Aggregation data is calculated for only the levels with an **Enable Aggregations** property of **Yes**. Custom is not valid for parent-child dimensions or dimensions with their **Changing** property set to **True**.</li></ul>
	All Caption	The name of the member in the (All) level.
	All Level	Indicates whether the dimension contains an (All) level. If the value is **Yes**, the (All) level is the top level of the dimension but is not displayed in the Cube Editor tree pane. The (All) level contains a single member whose cell value is the aggregate of cell values for all members in the next lower level.
	All Member Formula	The custom rollup formula for the (All) level. This formula is an MDX expression that determines the cell values for the All member and overrides the **Aggregate Function** properties of measures. For more information, see Custom Rollup Formulas and Custom Member Formulas.
	Allow	Indicates whether the members under a

	Duplicate Names	common parent can have the same name.
	Changing	Indicates whether the dimension is optimized for frequent changes. If the value is set to **True**, query performance may be slower. However, levels and members below the top level and above the bottom level can be added, moved, and deleted. This eliminates or reduces subsequent processing requirements, and also minimizes interruptions of end users' access to the cubes that include that dimension. Several conditions must exist before the **Changing** property can be set to **True**. For more information, see Changing Dimensions.
	Data Member Caption Template	Controls the names of data members when the **Members With Data** property of the dimension is set to **Nonleaf data visible**. When you type a value that includes an asterisk (*), the name of each data member will be the value with the asterisk replaced by the name of the parent member. The **Data Member Caption Template** property is available only for parent-child dimensions.
	Data source	The data source that contains the dimension table(s).
	Default Member	The member that slices the datasets returned by queries when the dimension is not displayed on an axis and no slicing member in the dimension is specified. If no default member is specified, and the **All Level** property of the dimension is **Yes**, the member indicated by the **All Caption** property is the default member. If no default member is specified, and the **All Level** property of the dimension is **No**, an arbitrary member of the highest level is the default member.

	Depends on Dimension	For virtual dimensions, the dimension that supplies the member properties or columns on which the levels of the virtual dimension are based. For dimensions that are not virtual, the dimension according to which aggregation design is optimized. A dimension in the value of this property is advantageous when the cross product of the two dimensions' members results in a significant percentage of combinations that cannot coexist. For example, the **Depends on Dimension** property of a Customer Gender dimension is Customers. Fifty percent of the combinations resulting from the cross product of the dimensions' lowest-level members cannot coexist because a customer can have only one gender. For more information, see Dependent Dimensions.
	Description	The description of the dimension.
	Member Keys Unique	Indicates whether member keys are unique within the dimension. If this property is set to **True**, the **Member Keys Unique** property for every level in the dimension is set to **True**. If the value of this property is changed, you must reprocess the current cube using the **Full Process** option. For more information, see Processing Cubes.
	Member Names Unique	Indicates whether member names are unique within the dimension. If the value is **True**, internal member names omit qualifying level names and member names. These internal member names are used in MDX expressions. If this property is set to **True**, the **Member Names Unique** property for every level in the

		dimension is set to **True**.
	Members With Data	Indicates whether nonleaf members are allowed to have associated fact table data. If they are allowed, this property also indicates whether children of nonleaf members are created to display this data. Valid values are: • Leaf members only: Only leaf members can have associated fact table data. If a nonleaf member has associated fact table data, processing fails. This value is the default. • Nonleaf data hidden: Nonleaf members can have associated fact table data. This data is not represented among the descendants of the nonleaf member. Consequently, it might appear to end users that values aggregate incorrectly. • Nonleaf data visible: Nonleaf members can have associated fact table data. This data is represented among the descendants of the nonleaf members by the creation of a child for each nonleaf member. This child, called a data member, is a leaf member and has a value equal to the aggregate of its parent's associated fact table data. The data members' names are controlled by the **Data Member Caption Template** property. This property is available only for parent-child dimensions.

	Enable All Level Aggregations	Determines whether to consider the (All) level when designing aggregations to optimize performance.
	Name	The name of the dimension.
	Source Table Filter	A WHERE clause expression applied to the dimension table to limit the members in the dimension. For example, in the Store dimension supplied with Analysis Services, to include only the Canada member and its descendants, type: "store"."store_country" = 'Canada'
	Storage Mode	The type of storage for the dimension. If the value is **MOLAP** (multidimensional OLAP), the dimension data is stored in a multidimensional structure on the Analysis server. If the value is **ROLAP** (relational OLAP), the dimension data is the dimension table itself. **MOLAP** provides better performance and is recommended except for extremely large dimensions, that is, dimensions that have roughly 5 to 10 million members. In order to select **ROLAP**, the **Member Keys Unique** property of the lowest level must be **True**. Restrictions also apply to the dimension's **Aggregation Usage** property in all cubes in which the dimension is used. For more information, see Dimension Storage Modes.
	Type	The type of the dimension. Standard is the default. This property indicates to client applications the kind of information in the dimension.
	Virtual	Indicates whether the dimension is a virtual dimension, that is, one based on the tables and

		columns of another dimension. If this property is set to **True**, an (All) level is automatically created for the dimension.
	Visible	Indicates whether the dimension is visible when end users browse the cube.
	Write-enabled	Indicates whether the members of the dimension can be updated while administrators browse the dimension and while end users browse the cube. The only end users that can update a write-enabled dimension are those in cube roles granted read/write access to that dimension. Only parent-child dimensions can be write-enabled. For more information, see Write-Enabled Dimensions.
	Enable Real-Time Updates	Indicates whether the dimension supports real-time updates to underlying dimension tables.
Level	**Custom Members**	Indicates whether custom member formulas are used to determine cell values of the members. For more information, see Custom Rollup Formulas and Custom Member Formulas. To view the custom member formulas, right-click the dimension, click **Browse,** and then see the custom member formula pane.
	Custom Member Options	Indicates whether calculation options can be defined for custom members on this level and unary operators on the following level. Calculation options are stored in a column in the dimension table. To set this property to **True**, you must first set the **Custom Members** property to **True**.
	Custom Rollup Formula	An MDX expression that determines the members' cell values and overrides the **Aggregate Function** properties of measures. For more information, see Custom Rollup Formulas and Custom Member Formulas.

		Allows dimension tables to be linked to fact tables when the data types of the columns do not match but the contents represent the same values.
	Description	The description of the level.
	Disabled	Indicates whether the level is included in the cube. This property is not available for parent-child dimensions. A level whose **Disabled** property is **Yes** cannot be referenced explicitly in calculated members and other MDX expressions.
	Enable Aggregations	Indicates whether aggregations are calculated for the level. The value of this property can be changed only if the **Aggregation Usage** property of the dimension is **Custom**. The value of this property is always **No** for levels in virtual dimensions. This property is not available for parent-child dimensions.
	Grouping	Indicates whether the level contains member groups. Member groups can be used to satisfy the maximum limit of 64,000 members under a parent. To group members, create a new level immediately above and identical to the level that exceeds the limit, and then set the **Grouping** property of the new level to **Automatic**. For more information, see Member Groups. If the level is in a shared dimension, you must do this in Dimension Editor.
	Hide Member If	Determines which members are hidden from end users browsing the cube. Hidden members support ragged dimensions, which contain logical gaps in member lineage, by hiding the members that occupy the gaps. Valid values are:

		<ul><li>Never hidden: No members are hidden.</li><li>No name: Every member whose name is null or an empty string is hidden.</li><li>Parent's name: Every member with the same name as its parent is hidden.</li><li>Only child with no name: Every member that is an only child and whose name is null or an empty string is hidden.</li><li>Only child with parent's name: Every member that is an only child and has the same name as its parent is hidden.</li></ul>This property is not available for parent-child dimensions.
	Key Data Size	The size (in bytes) of the columns that store member keys in aggregations. Member keys are copied from the column specified in the **Member Key Column** property.
	Key Data Type	The data type of the columns that store member keys in aggregations. Member keys are copied from the column specified in the **Member Key Column** property.
	Level Naming Template	Determines the level names displayed to end users browsing the cube. This property is available only for parent-child dimensions. If the level is in a private dimension, click this property and click the edit (**...**) button to

		display the **Level Naming Template** dialog box.
	Level Type	The type of the level. Regular is the default. The following values are used only in dimensions whose **Type** property is **Time**: Years, Half-Years, Quarters, Months, Weeks, Days, Hours, Minutes, Seconds, and Time-Undefined. The **Level Type** property indicates to client applications the kind of information in the level. This property is not displayed for parent-child dimensions.
	Member Count	The number of members in the level at the time members were last estimated by Analysis Services, or a user-provided estimate of the member count. If the level is in a private dimension, you can update the member count by clicking **Count Dimension Members** on the **Tools** menu.
	Member Key Column	The column that contains the member keys. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Keys Unique	Indicates whether member keys are unique within the level. This property is not available for parent-child dimensions. If the value of this property is changed, you must reprocess the current cube using the **Full Process** option. For more information, see Processing Cubes.
	Member Name	The column that contains the member names, which are displayed to end users browsing the

	Column	cube. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Names Unique	Indicates whether member names are unique within the level. If the value is **True**, internal member names omit qualifying member names. These internal member names are used in MDX expressions.
	Name	The name of the level.
	Order By	The sort order for displayed members. You can sort by member name, member key, or any member property defined for the level.
	Parent Key Column	The column that contains the keys for the members' parents. This property is available only for parent-child dimensions.
	Root Member If	Indicates the criteria by which members of the highest level (ignoring the (All) level, if any) are identified. The following values are valid: • Parent is blank, self, or missing: A member is in the highest level if any of the following three criteria is met. • Parent is blank: A member is in the highest level if its value in the **Parent Key Column** is null or 0 (zero). • Parent is self: A member is in the highest level if its value in the **Parent Key Column** is equal to its value in the **Member Key Column**.

		<ul><li>Parent is missing: A member is in the highest level if its value in the **Parent Key Column** does not exist in the **Member Key Column**.</li></ul> This property is available for parent-child dimensions only.
	Skipped Levels Column	Indicates the column that contains the number of levels between a member and its parent, excluding the member and parent. Valid values in the column are 0 (zero) and positive integers. This property is available only for parent-child dimensions.
	Unary Operators	Enables unary operators or custom rollup operators that control how level members are aggregated into the value of their parent member. For more information, see Custom Rollup Operators. To view the custom member formulas, on the **View** menu click **Data**, and then see the value beside UNARY_COLUMN in the Member properties pane. The structure of the cube will not be valid if a cube has a measure with the **Distinct Count** aggregate function, or if you add custom rollup operators to a level of the cube.
	Visible	Indicates whether the level is visible to end users browsing the cube. To set this property to **False**, you must first set the **Member Keys Unique** property of all lower levels to **True**. The **Visible** property is not available for parent-child dimensions. Unlike other objects whose **Visible** property is **False**, a level cannot be explicitly referenced in calculated members and other MDX expressions.

Measure	**Aggregate Function**	The function used to aggregate measure values. Valid values are: • **Sum**: The measure value for a dimension member is calculated by adding the values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. • **Count**: The measure value for a dimension member is calculated by adding the number of values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. Null values are not counted. • **Min**: The measure value for a dimension member is the lowest of the values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants • **Max**: The measure value for a dimension member is the highest of the values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants.

		<ul><li>**Distinct Count**: The measure value for a dimension member is calculated by adding the number of different values in the measure's **Source Column** in rows where the foreign key value for the dimension equals the key value of the member or one of its descendants. A cube can contain only one measure with the **Distinct Count** aggregate function.</li></ul>If a cube has a measure with the **Distinct Count** aggregate function, the structure of the cube will not be valid if you add a custom rollup operator or expression to a level of the cube. Measure values for the intersections of members from different dimensions are also calculated using the selected aggregate function. For more information about aggregate functions, see Aggregate Functions. A multiple-partition cube can have multiple fact tables, so a measure can have multiple source columns. (Note that each source column must have the same column name.) In this case the aggregate function spans all the source columns.
	Data Type	The data type of the columns that store measure values in aggregations.
	Description	The description of the measure.
	Display Format	The format of the measure values displayed to end users browsing the cube. For more information, see Display Formats.
	Name	The name of the measure. The name is displayed to end users browsing the cube.

	Source Column	In the fact table of the default partition of the cube, the column that stores the measure values before they are aggregated.
	Visible	Indicates whether the measure is visible to end users browsing the cube.
Member property	**Caption**	The caption used to display the member property.
	Data Size	The maximum number of characters allowed in the column that stores the member property values. Allows dimension tables to be linked to fact tables when the data types of the columns do not match but the contents represent the same values.
	Data Type	The data type of the column that stores the member property values. Allows dimension tables to be linked to fact tables when the data types of the columns do not match but the contents represent the same values.
	Description	The description of the member property.
	Language	The language used to display the member property.
	Name	The name of the member property.
	Source Column	The column that stores the values of the member property. This column must be in one of the dimension tables for the dimension that contains the member property.
	Type	Indicates to client applications the type of information in the member property values.
	Visible	Indicates whether the member property is visible to end users browsing the cube.
Named set	**Name**	The name of the named set.
	Value	The MDX expression that defines the named set.

Schema Tab (Cube Editor Schema View)

Use the **Schema** tab to display the cube's fact table with a yellow title bar and all dimension tables with blue title bars. Joins are indicated by lines that connect the key columns between the tables.

In the **Schema** tab, you can organize the structure of a cube in a graphical display, add measures and shared dimensions, and create private dimensions.

To view all columns in a table, lengthen the table window vertically, or use the scroll bars. You can also widen the table window to view long column names.

Note Changing the structure of a cube causes the results of previous cube processing to be invalidated. After a cube's structure has changed, you must process the cube to re-create its data.

Working in the Schema Tab

This section lists the actions that can be performed in the **Schema** tab. Many actions can be performed in multiple ways. For example, an action can be accomplished by a drag-and-drop procedure, or by right-clicking an object and then clicking an item on the shortcut menu.

To add a new table

- Right-click anywhere in the **Schema** tab, and then click **Insert Tables**. A join using the key column is automatically created between this new dimension table and the fact table.

To browse the data in a table

- Right-click the title bar of the table, and then click **Browse Data**.

To create a join between two tables

- Select the key column name in the first table, and then drag it to the corresponding key column name in the second table.

To remove a join between two tables

- Right-click the line depicting the join, and then click **Remove**.

To remove a dimension table

- Right-click the title bar of the table, and then click **Remove**.

To replace the fact table

- Right-click the title bar of the table you want to replace, and then click **Replace Fact Table**.

The schema is refreshed with the new fact table. Joins with the existing dimension tables are constructed from the common key columns in the corresponding tables.

When you replace the fact table, if the cube contains multiple partitions, the fact tables for the partitions are not automatically changed. (If the cube contains only one partition, the fact table for the partition is changed to match the new fact table for the cube.) To change a partition's fact table, in the Analysis Manager tree pane, right-click the partition, click **Edit**, and then use the Partition Wizard.

To change the alias of a table

- Right-click the title bar of the table, and then click **Change Alias**.

  Note If the table is the source table of a dimension that is used to define a virtual dimension, this capability is not available.

To add a new dimension

You can add a new cube dimension from any column contained in a table.

- Right-click the column, and then click **Insert as Dimension**.

  -or-

- Double-click the column. In the **Map the Column** dialog box, click **Dimension**, and then click **OK**.

  -or-

- Drag the column name from the **Schema** tab to the Dimensions folder.

Note If you add or delete dimensions in your cube, you must redesign the aggregations and reprocess the cube.

To add a new measure

You can add a new measure from any numeric or time column contained in the fact table. Each cube must contain at least one measure.

- Right-click the column, and then click **Insert As Measure**.

  -or-

- Double-click the column. In the **Map the Column** dialog box, click **Measure**, and then click **OK**.

  -or-

- Drag the column name from the **Schema** tab to the Measures folder.

  Note If you add or delete measures in your cube, you must redesign the aggregations and reprocess the cube.

Cube Processing Settings Dialog Box

Use the **Cube Processing Settings** dialog box to specify settings for processing a cube. Settings in this dialog box are used whenever a cube or one of its partitions is processed.

This dialog box appears when, in the **Process a Cube** dialog box, you click **Settings**. It also appears when, in the **Analysis Services Processing Task** dialog box, you click **Cube Settings**.

Important Cube or partition processing options are stored with the meta data for the cube or partition. Changes to the settings for these options replace previous settings for the cube or partition, regardless of whether the options are set by Analysis Manager, Data Transformation Services (DTS) tasks, or Decision Support Objects (DSO) applications.

Options

After all aggregations are calculated

Select to make data available for browsing only after all aggregations have been computed. This is the default option.

Immediately after data is loaded

Select to make data available for browsing after it has been loaded but before all aggregations have been computed. The new data is available for querying sooner, but the Analysis server may not be as responsive to query execution.

Stop processing after encountering missing dimension key errors

Select to stop processing if dimension key errors are encountered.

- **Processing will stop after**

  Specify the number of errors after which you want processing to stop. The default is zero.

Ignore all missing dimension key errors

Processing continues despite dimension key errors.

Log dimension key errors to a file

Select to provide a path to a text file. Universal Naming Convention (UNC) paths are recommended.

- **File path and name**

  Type the path into the **File path and name** dialog box.

- **Browse**

  Click to select a local or network location for the file.

See Also

Process a Cube Dialog Box

Processing Cubes

Analysis Services Processing Task Dialog Box

Cube Role Dialog Box

Use this dialog box to create or edit a cube role.

This dialog box is displayed when you perform any of the following actions in Cube Role Manager:

- Click **New** or **Edit**.

- In a checked row in the list, click in any column except **Role** or **Description**, and then click the edit (**...**) button. Depending on the column in which you click the edit (**...**) button, the appropriate tab is displayed, and the cursor is placed in the appropriate box.

The **Cube Role** dialog box has three tabs. For more information about the tabs, click a link in the following table.

To do this	See
View a summary of the role definition. This tab is visible only when you are editing an existing role.	Summary Tab (Cube Role Dialog Box)
Maintain the list of users and groups in the role. Changes in this tab propagate to the database role and cube roles with the same name as the edited cube role.	Membership Tab (Cube Role Dialog Box)
Control the role's access to the dimensions in the cube for which the role is defined.	Dimensions Tab (Cube Role Dialog Box)
Control the role's access to the cells in the cube for which the role is defined.	Cells Tab (Cube Role Dialog Box)
Enable or disable cube linking, drillthrough, and SQL queries for the role.	Options Tab (Cube Role Dialog Box)

Note The default values for a cube role are derived from the database role of the same name.

Options

The following options are displayed for all tabs.

Role name

View the name of the role. The name can be a maximum of 50 characters; it must begin with an alphabetical character. If you are editing an existing role, you cannot change the name of that role.

Description

View the description of the role. You can enter a maximum of 200 characters. This is not available if you are editing an existing role.

Enforce on

View or set the location of security enforcement: **Server** or **Client**. Server enforcement is more secure but may slow performance. Client enforcement generally provides better performance but increases the risk of unauthorized access to data on the client workstation.

If **Client** is selected, queries might be resolved partially or completely at the client workstation.

If **Server** is selected, queries are resolved entirely on the Analysis server or at the data source. User-defined functions stored exclusively on client workstations cannot be used.

Restore Defaults

Click to replace the entire cube role definition with the definition of the database role of the same name as the cube role.

Summary Tab (Cube Role Dialog Box)

Use this tab to view a summary of the cube role definition. This tab is read-only and appears only if you are editing a cube role.

Boxes

Membership

View the Microsoft® Windows NT® 4.0 or Windows® 2000 users and groups in the role.

Restricted Dimensions

View the cube's dimensions with a read permission or read/write permission of Fully Restricted or Custom.

To change the dimension security settings, use the **Dimensions** tab. For more information, see Dimensions Tab (Cube Role Dialog Box).

Cells

View information about current cell security settings.

To change the cell security settings, use the **Cells** tab. For more information, see Cells Tab (Cube Role Dialog Box).

Membership Tab (Cube Role Dialog Box)

Use this tab to maintain the list of Microsoft® Windows NT® 4.0 or Windows® 2000 users and groups in the role.

Note Changes in this tab propagate to the database role and cube roles with the same name as the edited cube role.

Options

Name

View the Windows NT 4.0 or Windows 2000 users and groups in the role.

Domain

View the domains of the users and groups in the role.

Add

View the **Add Users and Groups** dialog box so you can add new users or groups to the role.

Remove

Click to remove the selected user or group from the role.

Dimensions Tab (Cube Role Dialog Box)

Use this tab to control the role's access to dimensions in the cube for which the role is defined.

Dimension security is defined with permissions and rules. A dimension always has a read permission, but only write-enabled dimensions have read/write permissions and may be updated by end users. For each permission, you can select from various rules (described later in this topic).

Note Changes to a read/write permission propagate to the database role of the same name.

For more information about dimension security, see Dimension Security.

Options

Name

View the names of dimensions in the cube. Data mining dimensions are not displayed. Security for data mining dimensions is not supported in Analysis Services.

Permission

View or set the permissions associated with the dimensions. A dimension can have read or read/write permissions.

Option	Description
Read	Determines which dimension members the users in the role can view. All dimensions have a read permission.
Read/write	Determines which dimension members the users in the role can update. Only write-enabled dimensions have a read/write permission. If you allow access to a member in the read/write permission, it is viewable even if it is not accessible in the read permission.

For each displayed permission, select a rule.

Rule

View or set the rules associated with the displayed permissions. The following table describes the rules that are available for each permission.

Permission	Rule	Rule description
Read	Unrestricted	The role can view all members. This rule is the default.
	Fully Restricted	The role cannot view members. When users in the role browse the cube, they do not see the dimension.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be viewed. To access this dialog box, select **Custom**, and then in the **Custom Settings** column click the edit (**...**) button. For more information, see Defining Custom Rules for Dimension Security.
Read/write	Unrestricted	The role can update all members. This rule is available only if the rule for the read permission is Unrestricted.
	Fully Restricted	The role cannot update members. This rule is the default and is available only if the rule for the read permission is Unrestricted or Fully Restricted.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be updated. To access this dialog box, select **Custom**, and then in the **Custom Settings** column click the edit (**...**) button. This rule is available only if rule for the read permission is Unrestricted or Custom. For more information, see Defining Custom Rules for Dimension Security.

Custom Settings

View or set descriptions of custom rules. For other rules, this column is blank.

See Also

Write-Enabled Dimensions

Cells Tab (Cube Role Dialog Box)

Use this tab to control the role's access to cells in the cube for which the role is defined.

Cell security is defined with policies, permissions, and rules.

The default policy is Unrestricted Read. For a write-enabled cube, you can select a policy of Unrestricted Read/Write. For any cube, you can select a policy of Advanced, which allows you to select the cells that can and cannot be accessed.

You select or define rules for each displayed permission only with an Advanced policy. With an Advanced policy, a cube always has a read permission and a read contingent permission. However, only write-enabled cubes have read/write permissions and may be updated by end users. (A virtual cube has a read/write permission if one or more of its component cubes is write-enabled.) For each permission, you can select from various rules, which are described later in this topic.

For more information about cell security, see Cell Security.

Options

Cell security policy

Select from the following policies:

- **Unrestricted read**

  The role can view all cell values. This policy is the default.

- **Unrestricted read/write**

  The role can view and update all cell values. This policy is available only for write-enabled cubes. It is available for virtual cubes only if one or more of its component cubes is write-enabled.

- **Advanced**

The role can view and update only the cell values you specify in the permissions and rules.

- **Allow users to commit writeback changes**

  This option is available only for write-enabled cubes for which **Cell security policy** is Advanced. This option permits users to make actual changes to the writeback table when selected. If this option is not enabled, changes apply only to ad hoc analysis and are temporary.

Permission

View or set permissions. Permissions are used only if **Cell security policy** is Advanced. A cube role can have read, read contingent, or read/write permissions.

Option	Description
Read	Determines which cells the users in the role can view. Cube roles for all cubes have a read permission.
Read contingent	Determines which cells the users in the role can view, subject to the following condition: if a cell is specified in this permission and derived from other cells, it is viewable only if all the other cells are viewable. The other cells are deemed viewable if they are included in the read permission or included in the read contingent permission but not derived from other cells. If a cell is specified in the read contingent permission and not derived from other cells, it is viewable. The most common derived cells are for calculated members. For example, the calculated member Profit is derived from the measures Sales and Cost (Profit equals Sales minus Cost). If cells for Profit are specified in the read contingent permission, they are viewable only if cells for both Sales and Cost are viewable (that is, included in the read permission or included in the read contingent permission but not derived from other cells).

	If a cell is included in both the read and read contingent permissions, the read permission is enforced, but the read contingent permission is not. Cube roles for all cubes have a read contingent permission.
Read/write	Determines which cells the users in the role can update. A cube role has a read/write permission only if the associated cube is write-enabled or the associated virtual cube has one or more write-enabled, component cubes. If you allow access to a cell in the read/write permission, it is viewable even if it is not accessible in the read permission or read contingent permission. In this case the cell is viewable as if it were accessible in the read permission.

If **Cell security policy** is Advanced, for each displayed permission, select a rule.

Note Including derived cells in the read permission incurs the risk that end users might determine cell values they cannot view. For example, if Profit is included in the read permission, and cells for Cost are viewable, but cells for Sales are not, end users can determine Sales values by adding Profit and Cost values.

Rule

View or set rules. Rules are accessible only if **Cell security policy** is Advanced.

The following table describes the rules that are available for each permission.

Permission	**Rule**	**Rule description**
Read	Unrestricted	The role can view all cell values. This rule is the default.
	Fully Restricted	The role can view only the cell values specified in the read/write permission or read contingent permission, subject to its limitations described earlier in this topic.
	Custom	You can specify the cell values that are viewable and not viewable in the **Cube Cell**

		Security dialog box. To access this dialog box, select **Custom**, and then in the **Custom Settings** column click the edit (**...**) button.
Read contingent	Unrestricted	The role can view all cell values that are not derived from other cells. If a cell value is derived from other cells, it is viewable if all the other cells are included in the read or read/write permission.
	Fully Restricted	The role can view only the cell values specified in the read permission or read/write permission. This rule is the default.
	Custom	You can specify the cell values that are viewable and not viewable, subject to the limitations of the read contingent permission described earlier in this topic, in the **Cube Cell Security** dialog box. To access this dialog box, select **Custom**, and then in the **Custom Settings** column click the edit (**...**) button.
Read/write	Unrestricted	The role can update all cell values.
	Fully Restricted	The role cannot update cell values.
	Custom	You can specify the cell values that are updatable and not updatable in the **Cube Cell Security** dialog box. To access this dialog box, select **Custom**, and then in the **Custom Settings** column click the edit (**...**) button.

If the rules define a cell as not viewable, the cell itself is visible but its value is not.

Custom Settings

View descriptions of custom rules. For other rules, this column is blank.

See Also

Write-Enabled Cubes

Options Tab (Cube Role Dialog Box)

Use this tab to control the role's access to data in cells in the cube for which the role is defined.

Options

Allow drillthrough

Select to allow end users in the role to drill through to the source data of a cell. This ability also requires that drillthrough is enabled for the cube or at least one of its partitions. For more information, see Specifying Drillthrough Options.

This check box is cleared and read-only if the cube role is for a linked cube. Linked cubes do not support drillthrough.

Allow linking to this cube

Select to allow end users in the role to use this cube as the source cube for linked cubes. This check box is selected by default.

Allow sending SQL queries to this cube

Select to allow end users in the role to perform SQL queries to find cell values in the cube. This check box is selected by default.

See Also

Linked Cubes

Cube Role Manager

Use this tool to maintain cube roles.

A cube role applies to only a single cube. A cube role is created when you assign a database role to a cube by selecting the database role in Cube Role Manager. This action grants the role's users access to the cube. The name and default values of a cube role are derived from the selected database role. Some of these defaults can be overridden in the cube role. In addition, cube roles contain options such as cell security that are not contained in database roles.

In Cube Role Manager, each row with a selected check box in the list displays a cube role. Users in these roles can access the cube displayed in the title bar of Cube Role Manager. These roles can be maintained within Cube Role Manager.

Each unchecked row displays a database role not assigned to the cube. Users in these roles cannot access the cube unless they are also in one or more of the selected cube roles. The database roles cannot be maintained within Cube Role Manager. (To maintain them, use Database Role Manager.)

To grant users in a database role access to the cube, select the check box beside the database role. This action creates a cube role with the same name as the database role and removes the database role from the list.

CAUTION Clearing the check box beside a role deletes the cube role and all its settings, including cell security settings.

To deny users in a cube role access to the cube, clear the check box beside the cube role. This action deletes the cube role and adds to the list the database role with the same name as the cube role.

The list is sorted with cube roles followed by database roles. Each category is sorted alphabetically by the role names in the **Role** column.

Cube Role Manager appears when in the Analysis Manager tree pane, you right-click a cube and then click **Manage Roles**.

Options

View whether a role can access the cube displayed in the title bar of Cube Role Manager. Only checked roles can access the cube.

Role

View the role names.

Enforce on

Set the location of security enforcement: **Server** or **Client**. Server enforcement is more secure but may slow performance. Client enforcement generally provides better performance but increases the risk of unauthorized access to data on the client workstation.

If **Client** is selected, queries might be resolved partially or completely at the client workstation.

If **Server** is selected, queries are resolved entirely on the Analysis server or at the data source. User-defined functions stored exclusively on client workstations cannot be used.

To change the value, click it, click the edit (**...**) button, and then in the **Cube Role** dialog box, in the **Enforce on** box, select the new value.

Membership

View the Microsoft® Windows NT® 4.0 or Windows® 2000 users and groups in each role.

To change the membership of a cube role, click the cell where the role intersects **Membership**, click the edit (**...**) button, and then use the **Membership** tab of the **Cube Role** dialog box.

Note Changes in this tab propagate to the database role and cube roles with the same name as the edited cube role.

Restricted Dimensions

View the cube's dimensions with a read permission or read/write permission of Fully Restricted or Custom.

To access the dimension security settings for a cube role, click the cell where the role intersects **Restricted Dimensions**, click the edit (**...**) button, and then use the **Dimensions** tab of the **Cube Role** dialog box.

Cells

View the cell security policy. A lock icon indicates that the cell security policy of the role is Advanced.

If the **Cells** column is blank, the cell security policy is Unrestricted Read.

To access the cell security settings for a cube role, click the cell where the role intersects **Cells**, click the edit (**...**) button, and then use the **Cells** tab of the **Cube Role** dialog box.

Drillthrough

Select whether end users in the role can drill through to a cell's source data. This ability also requires drillthrough to be enabled for the cube or at least one of its partitions. For more information, see Specifying Drillthrough Options.

To change the value, click it, click the edit (**...**) button, and then on the **Options** tab of the **Cube Role** dialog box, select or clear the **Allow drillthrough** check box.

Description

View the description of each role. To change a description, click it, click the edit (**...**) button, and then in the **Cube Role** dialog box, in the **Description** box, type a new description.

Show

Click to limit the roles displayed in the list. You can limit by a user name or group name in the roles.

Roles containing users
Select to limit by a user name or group name in the roles. Type the user name or group name, or type the first part of a name, and then click the magnifying glass button.

Roles assigned to cubes
Select to limit by a cube to which the roles are assigned. Type the cube name, and then click the magnifying glass button.

New

Click to display the **Cube Role** dialog box so you can create a new cube role. When you create a new cube role, a database role with the same name and specifications is also created.

Edit

Click to display the selected cube role in the **Cube Role** dialog box, where you can edit the cube role.

Duplicate

Click to display the **Duplicate Role** dialog box, where you can supply a name for a new cube role based on the selected cube role, and to display the **Cube Role** dialog box, where you can define the new cube role. When you create a new cube role in this way, a database role with the same name and specifications is also created.

Test Role

Click to simulate the selected role by displaying Cube Browser, where you can browse the cube as if you are a user in the role. Use this button to test read permissions and read contingent permissions but not read/write permissions. Writeback is not supported by Cube Browser.

To test an end user's inclusion in multiple roles, select the roles and then click **Test Role**.

Custom Dimension Security Dialog Box

Use this dialog box to create a custom rule for dimension security.

You can specify a range of dimension levels that are accessible to the role. Within this range, you can also specify which dimension members are accessible.

Important If you limit access to levels or members by using the **Basic** or **Advanced** tab, consider selecting **Enable - Show visual totals** in the **Common** tab. If instead you use the default setting for visual totals (that is, **Disable - Do not show visual totals**), security exposures might be created. These exposures allow end users in the role to deduce values for members to which they are denied access. For more information, see Example 4, Example 6, Example 8, and Example 9 in Custom Rules in Dimension Security.

This dialog box is displayed when, in the **Dimensions** tab of the **Database Role** dialog box or **Cube Role** dialog box, you select a custom rule, and then, in the **Custom Settings** column, click the edit (**...**) button.

The **Custom Dimension Security** dialog box has three tabs. For more information about the tabs, click a link in the following table.

To do this	See
Define a custom rule by selecting levels from lists and members from a tree	Basic Tab (Custom Dimension Security Dialog Box)
Define a custom rule by supplying Multidimensional Expressions (MDX) Specify a range of dimension levels that are accessible to the role, and, within this range, specify which dimension members are accessible	Advanced Tab (Custom Dimension Security Dialog Box)
Specify options for visual totals and	Common Tab (Custom Dimension

select a default member	Security Dialog Box)

Options

The following options are displayed for all tabs.

Permission

View the permission to which the custom rule applies. This box is read-only.

Note If the custom rule applies to a read/write permission in a cube role, changes propagate to the database role of the same name.

Description

Enter the description of the custom rule.

Restore Defaults

Discards any changes you have made to a cube role.

OK

Click to close the dialog box. After you close the dialog box, save the custom rule by clicking **OK** in the **Edit a Database Role** dialog box or **Edit a Cube Role** dialog box (depending on whether you are working in a database role or cube role).

See Also

Custom Rules in Dimension Security

Basic Tab (Custom Dimension Security Dialog Box)

Use this tab to define a custom rule by selecting levels from lists and members from a tree.

This tab includes a procedure:

1. (Optional.) In the **Select visible levels** area, select levels.

2. In the **Select members** area, select a default setting for member selection, and then optionally select members.

The selections you make in this tab dynamically refresh the **Advanced** tab. For more information, see Advanced Tab (Custom Dimension Security Dialog Box).

Options

Top Level

Click to display the topmost level that can be accessed. Levels above the top level cannot be accessed. The top level must be above or the same as the bottom level. Members above the top level are not displayed in the **Members** box.

Bottom Level

Click to display the bottommost level that can be accessed. Levels below the bottom level cannot be accessed. The bottom level must be below or the same as the top level. Members below the bottom level are not displayed in the **Members** box.

Visible Levels

View the accessible levels depending on the values in the **Top Level** and **Bottom Level** boxes. The **Visible Levels** box is read-only.

Select all members

Click to select all members. You can then specify inaccessible members by clearing their check boxes in the **Members** box.

Note If you click **Select all members**, then clear one or more check boxes, and then decide you want to select all members, you must first click **Deselect all members** and then click **Select all members**.

Deselect all members

Click to clear the selection of all members. You can then specify accessible members by selecting their check boxes in the **Members** box.

Note If you click **Deselect all members**, then select one or more check boxes, and then decide you want to clear the selection of all members, you must first click **Select all members** and then click **Deselect all members**.

Members

View all members between the top level and bottom level, including those in the top level and bottom level. Selected members are accessible, and unselected members are not accessible.

Select a check box beside a member to allow access to it. This action also selects the member's descendants and ancestors that are visible in the **Members** box.

Clear a check box beside a member to deny access to it. This action also clears the selection of the member's descendants that are visible in the **Members** box.

Advanced Tab (Custom Dimension Security Dialog Box)

Use this tab to define a custom rule by supplying Multidimensional Expressions (MDX) statements.

You can specify a range of dimension levels that are accessible to the role, and, within this range, which dimension members are accessible.

This topic contains an example MDX expression for each option in this tab. For more information about these options and for more examples, see Custom Rules in Dimension Security.

The expressions you supply in this tab dynamically refresh the **Basic** tab if the expressions can be translated successfully. If they cannot be translated, you are notified with a message in the **Basic** tab. For more information, see Basic Tab (Custom Dimension Security Dialog Box).

Options

To supply MDX expressions for the following options, you can either type in the expression or click the edit (**...**) button to open MDX Builder.

The **Allowed Members** and **Denied Members** boxes are optional.

Top Level

Supply an MDX expression that represents the topmost accessible level. For example, in the Warehouse dimension included with Microsoft® SQL Server™ Analysis Services, to specify State Province as the top level, type the following expression:

```
[Warehouse].[State Province]
```

The top level must be above or the same as the bottom level. Members above the top level cannot be accessed even if they are included in the expression in the **Allowed Members** box.

Bottom Level

Supply an MDX expression that represents the bottommost accessible level. For example, in the Warehouse dimension, to specify City as the bottom level, type the following expression:

```
[Warehouse].[City]
```

The bottom level must be below or the same as the top level. Members below the bottom level cannot be accessed even if they are included in the expression in the **Allowed Members** box.

Allowed Members

Supply an MDX expression for the set of members that can be accessed. Descendants of these members can also be accessed unless they are above the top level, below the bottom level, or access to them is denied by the expression in the **Denied Members** box. The only nondescendants of these members that can be accessed are their ancestors at and below the top level.

For example, in the Warehouse dimension, to allow access to the WA and OR members, type the following expression:

```
{[Warehouse].[All Warehouses].[USA].[WA],
[Warehouse].[All Warehouses].[USA].[OR]}
```

Denied Members

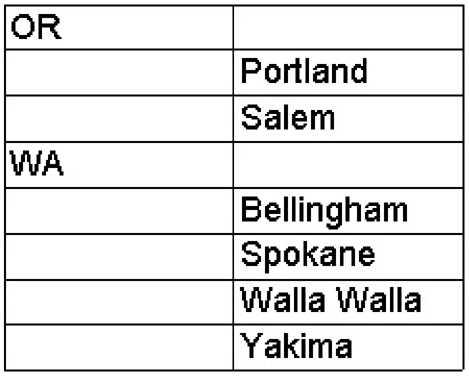
Supply an MDX expression for the set of members that cannot be accessed. Descendants of these members cannot be accessed unless access to them is allowed by the expression in the **Allowed Members** box. Nondescendants can be accessed unless they are above the top level or below the bottom level.

For example, in the Warehouse dimension, to deny access to the Seattle, Tacoma, and Bremerton members, type the following expression:

```
{[Warehouse].[All Warehouses].[USA].[WA].[Seattle],
[Warehouse].[All Warehouses].[USA].[WA].[Tacoma],
[Warehouse].[All Warehouses].[USA].[WA].[Bremerton]}
```

Combined Example

This example demonstrates the effect of using all the previous examples in this topic together in a read permission. If a user in the role issues an MDX query that projects the Warehouse dimension on the y-axis, the data set is returned, and the user fully expands the Warehouse dimension, it would appear as follows:



OR	
	Portland
	Salem
WA	
	Bellingham
	Spokane
	Walla Walla
	Yakima

Members in the (All), Country, and Warehouse Name levels are not visible because they are above the top level or below the bottom level. The only members of the State Province level that are visible are OR and WA because they are specified in the **Allowed Members** box. The only members of the City level that are visible are descendants of OR and WA that are not denied by the expression in the **Denied Members** box.

Common Tab (Custom Dimension Security Dialog Box)

Use this tab to specify options for visual totals and to define a default member.

The options for visual totals determine whether users see aggregated cube cell values that are calculated with values for:

- Only the members they can see (users see visual totals).

- All members (users see actual totals).

One option produces visual totals at and above a specified level, and actual totals below it.

If you select a default member, and a query is issued on a cube that includes the dimension, but the query does not project the dimension on an axis, by default the returned dataset is filtered (that is, sliced) by the default member. If you do not select a default member, the default member is determined by the **Default Member** property of the dimension, which is accessed in the properties pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private).

Options

Enable - Show visual totals

Displayed, aggregated cell values are calculated according to only the viewable members.

Note Visual totals cannot be enabled for a cube that contains a distinct count measure. For more information, see Using Aggregate Functions.

Custom - Show visual totals starting at the following level and above

To produce visual totals at and above a certain level, but to produce actual totals below it, select this option and, in the box below it, type an expression for the level in Multidimensional Expressions (MDX). Or, instead of typing, beside the box you can click the build (**...**) button to access MDX Builder, where you can select the level in the **Data** box and drag it to the **MDX expression** box. The MDX expression must name a level or resolve to a level.

For example, in the Geography dimension, to display visual totals at and above the Continent level, type:

```
[Geography].[Continent]
```

IMPORTANT This option creates one or more security exposures below the specified level when it allows end users to calculate values for a denied member by combining values for allowed members. For example, in the Geography dimension, if France has two children, Northern France and Southern France, and only Northern France is denied, users in the role can calculate values for Northern France by subtracting values for Southern France from values for France.

Disable - Do not show visual totals

Displayed, aggregated cell values are calculated according to all members, whether they are viewable or not. This option is the default for visual totals.

IMPORTANT This option creates one or more security exposures when it allows end users to calculate values for a denied member by combining values for allowed members. For example, in the Geography dimension if France has two children, Northern France and Southern France, and only Northern France is denied, users in the role can calculate values for Northern France by subtracting values for Southern France from values for France.

Define default member and specify using MDX

To select a default member, select this check box and in the box below it type an MDX expression for the default member. Or, instead of typing, beside the box you can click the edit button (**...**) to access MDX Builder, where you can select the default member in the **Data** box and drag it to the **MDX expression** box.

For example, in the Geography dimension to select the France member, type:

[Geography].[All Geography].[Europe].[France]

Data Mining Model Browser

Use this tool to provide visualization of data mining model content in an easily understandable format. It consists of several panes, as shown here.

Data Mining Model Browser appears when you do one of the following:

- In the Analysis Manager tree pane, right-click a data mining model, and then click **Browse**.

- In the Analysis Manager tree pane, click a data mining model, and then, on the **Action** menu, click **Browse**.

Data Mining Model Browser has five areas. For more details about the areas, click a link in the following table.

To do this	See
Change the view of the content detail pane and select prediction trees.	Toolbar (Data Mining Model Browser)
View the structure of the mining model. View the nodes that define the data mining model content, and the rules used by each node.	Content Detail Pane (Data Mining Model Browser)
Browse the data mining model content, especially on complex structures. View data density of the entire data mining model. Change the view of the content detail pane.	Content Navigator Pane (Data Mining Model Browser)
View the attributes, including distribution data, and node path for the data mining model node selected	Attributes Pane (Data Mining Model Browser)

in the content detail pane.	
Change the attribute used to color code the nodes in the data mining model.	Legend Pane (Data Mining Model Browser)
Use keyboard shortcuts to navigate the browser.	Keyboard Shortcuts (Data Mining Model Browser)

See Also

Viewing with Data Mining Model Browser

Toolbar (Data Mining Model Browser)

Use the Data Mining Model Browser toolbar to change the view of the content detail pane and select prediction trees.

Tool	Description
Zoom In	Zoom in to the data mining model content in the content detail pane.
Zoom Out	Zoom out of the data mining model content in the content detail pane.
Prediction Tree list	A data mining model can have multiple prediction trees, depending on the model structure and the data mining provider. The Prediction Tree list allows you to view the prediction trees for each model individually.

See Also

Content Detail Pane (Data Mining Model Browser)

Viewing with Data Mining Model Browser

Data Mining Model Browser

Content Detail Pane (Data Mining Model Browser)

Use the content detail pane to view the structure of the mining model. The content detail pane shows the structure of the data mining model by displaying the nodes that define the data mining model content, as well as the rules used by each node, if applicable. Each data mining model node is shaded, based on the data density of that node, according to the legend provided at the bottom of Data Mining Model Browser. Generally speaking, the lighter the color, the lower the data density for the selected attribute. The positioning of the nodes also provides information about the data mining model. The nodes are shown in left-to-right order, based on the ranking of factors within the data mining model. The farther down the tree a split occurs, the less influence the fact that causes the split has in the data mining model in general.

Relationships between nodes are displayed as lines connecting nodes and are referred to as node paths. The attributes pane shows the rules used to follow a given node path when a node is selected in this pane.

Nodes can be selected in the content detail pane, so that the attributes, distribution, and rules of that node can be viewed in the attributes pane. To select a data mining model node, click it; a highlight border appears around the selected node as shown in the diagram.

Right-click on the content detail pane to display a menu with the following two options.

Menu option	Description
Zoom In	Zoom in on the data mining model content.
Zoom Out	Zoom out from the data mining model content.

See Also

Attributes Pane (Data Mining Model Browser)

Viewing with Data Mining Model Browser

Data Mining Model Browser

Content Navigator Pane (Data Mining Model Browser)

Use the content navigator pane to easily browse the data mining model content, especially on complex structures. The content navigator pane also provides a concise view of the data density for the entire data mining model, by shading the mining model content according to the legend provided at the bottom of Data Mining Model Browser.

The content detail area represents the portion of the data mining model currently displayed in the content detail pane. Clicking on a different area in the content navigator pane can move this area, and it changes the view of the content detail pane.

See Also

Content Detail Pane (Data Mining Model Browser)

Viewing with Data Mining Model Browser

Data Mining Model Browser

Attributes Pane (Data Mining Model Browser)

Use the attributes pane to view the attributes, including distribution data, and node path for a data mining model node selected in the content detail pane.

Tool	Description
Node Attribute Sets	Use this to change the attribute set used to determine the shading for nodes in the content detail and content navigator panes. This appears for all cluster models and for any tree model based on multiple attribute sets.
Node information tab	For a node attribute set, a set of totals and distributions can be viewed; this set is limited to 20 attributes. Use the **Totals** tab to view the case totals for the selected node and the **Histogram** tab to view a distribution graph for the cases pertaining to the selected node. Double-click a column heading to change the order in which the attributes are sorted.
Node path	Use this to view the rules to be satisfied in order to reach the selected node. All the rules needed to reach a node are displayed in this area, along a given node path.

Note Due to the data mining algorithms used in Microsoft® SQL Server™ 2000 Analysis Services, it is possible for the attributes pane to display probabilities greater than 0% when there are no cases for an attribute.

See Also

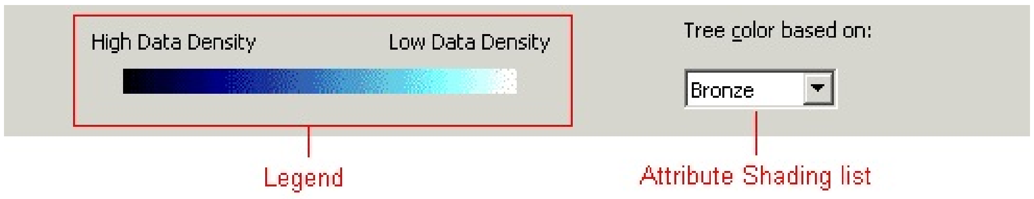
Content Detail Pane (Data Mining Model Browser)

Viewing with Data Mining Model Browser

Data Mining Model Browser

Legend Pane (Data Mining Model Browser)

Use the legend pane to view and change the attribute used to shade the nodes in the content detail pane.



The shading is determined on the number of cases for the specified attribute relative to the total number of cases evaluated for the tree or node. For example, if the total number of cases evaluated for a given node was 80, and the number of cases associated with a specified attribute was 40, then the shading for the node would be in the middle of the range of shades indicated by the legend.

Tool	Description
Legend	Displays the range of colors used to represent high and low data density of nodes. If an attribute is selected with the Attribute Shading list, the legend is used to represent the high and low attribute probability of nodes for the selected attribute.
Node attribute sets	Use this to change the attributes set used to determine shading for nodes in the content detail and content navigator panes. This appears for all cluster models and for any tree model based on multiple attribute sets.
Attribute Shading List	Use this to change the attribute used to determine the shading for each node in the content detail pane and content navigator pane.

See Also

Content Detail Pane (Data Mining Model Browser)

Content Navigator Pane (Data Mining Model Browser)

Viewing with Data Mining Model Browser

Data Mining Model Browser

Keyboard Shortcuts (Data Mining Model Browser)

This table describes the keyboard shortcuts available in Data Mining Model Browser. You can use shortcuts to access all parts of the browser except the content navigator pane.

Some sections support additional shortcuts. After a section has focus, you can use shortcuts to navigate within the section.

Activity	Shortcut
Move the focus to the next item. Navigation progression moves from the top left towards the bottom right of the screen. You can use the TAB key to access the prediction tree, content detail pane, attributes pane, node path, attribute shading list and Help. When the focus is on the content detail pane, you can use arrow keys to navigate the model.	TAB (SHIFT+ TAB for reverse order)
Toggle the focus between the content detail and attributes panes.	F6
When the focus is on the content detail pane, right-click in the pane to access the **Zoom In** and **Zoom Out** menu commands.	SHIFT+F10
Zoom out, when focus is on the content detail pane.	CTRL+ -
Zoom in, when focus is on the content detail.	CTRL+ +
Move the focus to the prediction tree pane, and then use arrow keys to navigate the list and the ENTER key to select a different value.	ALT+ P
Move the focus to the attribute shading list, and then use arrow keys to navigate the list and the ENTER key to select a different value.	ALT+ O
Move the focus to the node path pane, and then use arrow keys to navigate the text.	ALT+ D

Move the focus to the attributes pane, and then use the arrow keys to navigate between the **Totals** and **Histograms** tabs and to scroll tab contents.	ALT+ U
Copy text, when the focus is on an attribute.	CTRL+ C
Select all, when the focus is on attributes.	CTRL+ A
Open Help.	ALT+ H

See Also

Content Detail Pane (Data Mining Model Browser)

Content Navigator Pane (Data Mining Model Browser)

Viewing with Data Mining Model Browser

Data Mining Model Browser

Data Mining Prediction Query Task Dialog Box

Use this dialog box to add a Data Transformation Services (DTS) task that creates a prediction query and an output table from a data mining model object defined in Microsoft® SQL Server™ 2000 Analysis Services. For more information about using DTS to create prediction queries, see Creating Predictions Using Data Transformation Services.

This dialog box is displayed when you perform either of the following actions in DTS Designer:

- Drag the icon for the Data Mining Prediction Query task from the **Task** toolbar to the design sheet.

- Right-click a Data Mining Prediction Query task, and then click **Properties**.

- The **Data Mining Prediction Query Task** dialog box has three tabs. For more information about the tabs, click a link in the following table.

To do this	See
Specify an Analysis server, a database, and a data mining model for the Data Mining Prediction Query task	Mining Model Tab (Data Mining Prediction Query Task Dialog Box)
Specify a prediction query and a data source for the query and its output table	Query Tab (Data Mining Prediction Query Task Dialog Box)
Specify a data source for the output table of the prediction query	Output Tab (Data Mining Prediction Query Task Dialog Box)

Options

Name

Enter a name for the task.

Description

(Optional.) Enter a description for the task. This description becomes the label on the design sheet.

Mining Model Tab (Data Mining Prediction Query Task Dialog Box)

Use this tab to specify an Analysis server, a database, and a data mining model for the Data Mining Prediction Query task.

Options

Server

Specify an Analysis server name matching the computer name on the network that contains the mining model. Do not use universal naming convention (UNC) or network paths.

Database

Select an existing database on the specified Analysis server.

Data Mining Models

Click the name or icon of a mining model to select it for the task.

Details

View a description of the selected mining model. The description includes its algorithm and processing status.

Query Tab (Data Mining Prediction Query Task Dialog Box)

Use this tab to specify a prediction query and a data source for the query and its output table.

Options

Input data source

Enter a valid Microsoft® ActiveX® Data Objects (ADO) connection string to the data source of the input query, or click the edit (**...**) button to display the **Data Link Properties** dialog box, where you can build the connection string.

Prediction query

Type the syntax for the query, or click **New Query** to display Prediction Query Builder, where you can build the query syntax. Syntax must conform to the OLE DB for Data Mining specification. For more information about the OLE DB for Data Mining specification, see the Microsoft OLE DB Web page at the Microsoft Web site.

New Query

Click to start Prediction Query Builder.

See Also

Prediction Query Builder

Output Tab (Data Mining Prediction Query Task Dialog Box)

Use this tab to specify a data source for the output table of the prediction query.

Options

Output data source

Enter a valid Microsoft® ActiveX® Data Objects (ADO) connection string to the data source for the output table, or click the edit (**...**) button to display the **Data Link Properties** dialog box, where you can build the connection string.

Output table

Enter a name for the output table of the prediction query.

Database Dialog Box

Use the **Database** dialog box to create a new Microsoft® SQL Server™ 2000 Analysis Services database or change the description of an existing database.

This dialog box appears when you right-click a database or a server in the Analysis Manager tree pane and then click **New Database**. It also appears when you right-click a database and then click **Edit**.

Options

Database name

Type a name for your database. This option is not available if you are editing an existing database.

Description

Type a short description of your database.

Database Role Dialog Box

Use this dialog box to create or edit a database role.

This dialog box is displayed when you perform any of the following actions in Database Role Manager:

- Click **New** or **Edit**.

-or-

- In a row in the list, click in any column except **Role**, and then click the edit (**...**) button. Depending on the column in which you click the edit (**...**) button, the appropriate tab is displayed, and the cursor is placed in the appropriate box.

The **Database Role** dialog box has four tabs. For more information about the tabs, click a link in the following table:

To do this	See
Maintain the list of users and groups in the role.	Membership Tab (Database Role Dialog Box)
Grant or deny access to the cubes in the database.	Cubes Tab (Database Role Dialog Box)
Grant or deny access to the data mining models in the database.	Mining Models Tab (Database Role Dialog Box)
Control the role's access to dimensions in the database.	Dimensions Tab (Database Role Dialog Box)

Options

The following options are available for all tabs.

Role name

Enter the name of the role. You can enter a maximum of 50 characters; the name must begin with an alphabetical character. If you are editing an existing role, this box displays the name of that role, and you cannot change it.

Description

Enter the description of the role. You can enter a maximum of 200 characters.

Enforce on

Select the location of security enforcement: **Server** or **Client**. Server enforcement is more secure but may slow performance. Client enforcement generally provides better performance but increases the risk of unauthorized access to data on the client workstation.

If **Client** is selected, queries might be resolved partially or completely at the client workstation.

If **Server** is selected, queries are resolved entirely on the Analysis server or at the data source. User-defined functions stored exclusively on client workstations cannot be used.

Membership Tab (Database Role Dialog Box)

Use this tab to maintain the list of users and groups in the role.

Note Changes in this tab are propagated to all cube roles with the same name as the database role.

Options

Name

View the Microsoft® Windows NT® 4.0 or Windows® 2000 users and groups in the role.

Domain

View the domains of the users and groups in the role.

Add

View the **Add Users and Groups** dialog box so you can add new users or groups to the role.

Remove

Click to remove the selected user or group from the role.

Cubes Tab (Database Role Dialog Box)

Use this tab to grant or deny access to the cubes in the database.

To grant access to a cube, select the check box beside it. This action also creates a cube role with the same name as the database role.

Caution Clearing the check box beside a cube deletes the associated cube role and all its settings, including cell security settings.

To deny access to a cube, clear the check box beside it. This action also deletes the cube role that is associated with the cube and has the same name as the database role.

Options

🟨

This column determines whether a cube can be accessed by users in the role. Only checked cubes can be accessed**.**

Cube name

View the names of the cubes in the database.

Check All

Click to select all displayed cubes.

Clear All

Click to select all displayed cubes.

Mining Models Tab (Database Role Dialog Box)

Use this tab to grant or deny access to the data mining models in the database.

To grant access to a mining model, select the check box beside it. This action also creates a mining model role with the same name as the database role.

Note Changes in this tab are propagated to all mining model roles with the same name as the database role.

To deny access to a mining model, clear the check box beside it. This action also deletes the mining model role that is associated with the mining model and has the same name as the database role.

Options

Only mining models with check boxes that are selected can be accessed by users in the role.

Mining model name

View the names of the mining models in the database.

Check All

Click to select all displayed mining models.

Clear All

Click to select all displayed mining models.

Dimensions Tab (Database Role Dialog Box)

Use this tab to control the role's access to dimensions in the database.

Dimension security is defined with permissions and rules. A dimension always has a read permission, but only write-enabled dimensions have read/write permissions and may be updated by end users. For each permission you can select from various rules.

Note The dimension security settings in a database role can be overridden in cube roles of the same name.

For more information about dimension security, see Dimension Security.

Options

Name

View the names of dimensions in the database.

Private dimensions and data mining dimensions are not displayed. To define security for private dimensions, use the **Dimensions** tab of the **Cube Role** dialog box. Security for data mining dimensions is not supported in Analysis Services.

Permission

View the permissions associated with the dimensions. A dimension can have read or read/write permissions.

Option	Description
Read	Determines which dimension members the users in the role can view. All dimensions have a read permission.
Read/write	Determines which dimension members the users in the role can update. Only write-enabled dimensions have a read/write permission. If you allow access to a member in the read/write permission, it is viewable even if it is not accessible in the read permission.

For each displayed permission, select a rule.

Rule

View the rules associated with the displayed permissions. The following table describes the rules that are available for each permission.

Permission	Rule	Rule description
Read	Unrestricted	The role can view all members. This rule is the default.
	Fully Restricted	The role cannot view members. When users in the role browse a cube that includes the dimension, they do not see it.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be viewed. To access this dialog box, select Custom, and then in the **Custom Settings** column, click the edit (**...**) button.
Read/write	Unrestricted	The role can update all members. This rule is available only if the rule for the read permission is Unrestricted.
	Fully Restricted	The role cannot update members. This rule is the default and is available only if the rule for the read permission is Unrestricted or Fully Restricted.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be updated. To access this dialog box, select Custom, and then in the **Custom Settings** column, click the edit (**...**) button. This rule is available only if the rule for the read permission is Unrestricted or Custom.

Custom Settings

View descriptions of custom rules. For other rules, this column is blank.

See Also

Defining Custom Rules for Dimension Security

Write-Enabled Dimensions

Database Role Manager

Use this dialog box to create and maintain database roles or to set values for database roles assigned to data mining models.

Each row in the list displays a database role. The list is sorted alphabetically by the role names in the **Role** column.

Each database role defines a set of users and groups and the access they share. A database role can be assigned to any cube (including virtual and linked cubes) and data mining models in the database. This action grants users in the role access to the cube or mining model. It also creates a cube role or mining model role with the same name as the database role.

Note Some values in a database role can be overridden in cube roles of the same name. However, database role values cannot be overridden for mining model roles of the same name, because you cannot edit role properties in Mining Model Role Manager.

A cube role applies to only a single cube. The default values for a cube role are derived from the database role of the same name. Some of these defaults can be overridden in the cube role. In this case, the specifications in the database role are not the specifications used when users in the role access the cube. To determine the specifications that are used, see the cube role. Cube roles are maintained in Cube Role Manager.

Database Role Manager appears when, in the Analysis Manager tree pane, you right-click a database and then click **Manage Roles**.

Options

Role

View the database role names.

Enforce on

View the location of security enforcement: **Server** or **Client**. Server enforcement is more secure but may slow performance. Client enforcement generally provides better performance, but it may allow users to gain unauthorized access to data on the client workstation.

If **Client** is selected, queries might be resolved partially or completely at the client workstation.

If **Server** is selected, queries are resolved entirely on the Analysis server or at the data source. User-defined functions stored exclusively on client workstations cannot be used.

To change the value, click it, click the edit (**...**) button, and then in the **Database Role** dialog box, in the **Enforce on** box, select the new value.

Membership

View the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 users and groups in each database role.

To change the membership of a database role, click the cell where the role intersects **Membership**, click the edit (**...**) button, and then use the **Membership** tab of the **Database Role** dialog box.

Note Changes in this tab are propagated to all cube roles with the same name as the database role.

Cubes & Mining Models

View the cubes and mining models to which each database role has access.

To change the cubes or mining models a database role can access, in the cell where the role intersects **Cubes & Mining Models**, click the edit (**...**) button to display the **Database Role** dialog box. Next, use the **Cubes** or the **Mining Models** tab.

Restricted Dimensions

View the database's dimensions with a read permission or read/write permission of Fully Restricted or Custom.

To access the dimension security settings for a database role, click the cell where the role intersects **Restricted Dimensions**, click the edit (**...**) button, and then use the **Dimensions** tab of the **Database Role** dialog box.

Description

View the description of each database role.

To change a description, click it, click the edit (**...**) button, and then in the **Database Role** dialog box, in the **Description** box, type a new description.

Show

Click to limit the database roles displayed in the list. You can limit by a user name or group name in the roles or by a cube to which the roles are assigned.

To limit by a user name or group name in the roles, next to **Show**, in the first box, select **Roles containing users**. In the next box, type the user name or group name, or type the first part of a name. Click the magnifying glass button.

To limit by a cube to which the roles are assigned, next to **Show**, in the first box, select **Roles assigned to cubes**. In the next box, type the cube name. Click the magnifying glass button.

New

Click to display the **Database Role** dialog box, where you can create a new database role.

Edit

Click to display the selected database role in the **Database Role** dialog box, where you can edit the database role.

Duplicate

Click to display the **Duplicate Role** dialog box, where you can supply a name for a new database role based on the selected database role, and the **Database Role** dialog box, where you can define the new database role.

Delete

Click to delete the selected database role.

Define Custom Member Column Dialog Box

Use this dialog box to:

- Enable or disable the use of custom member formulas.

- Create or select a column to store custom member formulas. This column is created in or selected from the same dimension table that stores the members to which the custom member formulas apply.

Custom member formulas are expressions in Multidimensional Expressions (MDX) that determine the cell values associated with members and override the aggregate functions of measures. For write-enabled dimensions, custom member formulas are created in the custom member formula pane of Dimension Editor (if the dimension is shared) or Dimension Browser. (This action requires that the write-enabled dimension is included in a cube that was processed since the dimension last changed.) For other dimensions, they are inserted into the dimension table using a tool other than Analysis Manager.

For more information about custom member formulas, see Custom Rollup Formulas and Custom Member Formulas.

For more information about related procedures, see Creating Custom Member Formulas.

The **Define Custom Member Column** dialog box appears when, in the tree pane of Dimension Editor or Cube Editor, you select a level, and in the **Advanced** tab of the properties pane click the value beside **Custom Members**, then click the edit (**...**) button.

Options

Enable Custom Members

Select to enable the use of custom member formulas. This action sets the level's **Custom Members** property to **True**. Cell values for members with custom member formulas will be calculated according to the custom member formulas.

Clear to disable the use of custom member formulas. This action sets the level's **Custom Members** property to **False**. Custom member formulas will be ignored; cell values for members will be calculated according to the measures' aggregate functions.

Create a new column

Click to create a new column in which to store custom member formulas. Specify the column name in the **New column name** box, which appears only when **Create a new column** is selected.

Use an existing column

Click to select an existing column in which to store custom member formulas. Select the column in the **Existing column** box, which appears only when **Use an existing column** is selected.

Note If you select an existing column, its contents will be overwritten by the custom member formulas.

New column name

Type the name of the column that will store custom member formulas. This option is available only if you select **Create a new column**.

A dimension table can have multiple columns (one per level) that store custom member formulas. Therefore, it is recommended that the new column name identify the column that stores the members to which the custom member formulas apply. For example, if the members are stored in the **store_country** column, type:

```
store_country_custom_member_formula
```

Existing column

Select the column that will store custom member formulas. This option is available only if you select **Use an existing column**.

OK

Click to temporarily save your changes and close the dialog box. To

permanently save your changes, after this dialog box closes, on the **File** menu, click **Save**.

Define Custom Member Options Dialog Box

Use this dialog box to:

- Enable or disable the use of custom member options.

- Create or select a column to store custom member options. This column is created in or selected from the same dimension table that stores the members to which the custom member options apply.

Custom member options are cell properties defined for custom members. Custom member options can be used to change the font, display format, and other characteristics of the cells for the specified custom member. For more information about cell properties, see Using Cell Properties.

This property is available only when the **Custom Members** property for the level is set to **True**. This property is ignored for any member not defined as a custom member. In other words, no Multidimensional Expressions (MDX) expression has been defined for a given member in the custom member column.

The **Custom Member Options** property accepts a column reference containing, for each member, a comma-delimited list of cell properties. The cell properties are represented as string expressions. For more information, see Custom Member Options in Using Cell Properties.

The **Define Custom Member Options** dialog box appears when, in the tree pane of Dimension Editor (if the dimension is shared) or Cube Editor (if the dimension is private), you select a level, and in the **Advanced** tab of the properties pane click the value beside **Custom Member Options**, and then click the edit (**...**) button. This button is available only when the **Custom Members** property is set to **True**.

Options

Enable Custom Member Options

Select to enable the use of custom member options to determine the cell properties of custom members. This action sets the **Custom Member Options** property of the level to **True**.

Clear to make the use of custom member options unavailable. This action sets the **Custom Member Options** property of the level to **False**. Custom member options will be ignored.

Create a new column

Click to create a new column in which the cell properties for custom members will be stored. Specify the column name in the **New column name** box, which appears only when **Create a new column** is selected.

Use an existing column

Click to select an existing column in which to store cell properties for custom members. Select the column in the **Existing column** box, which appears only when **Use an existing column** is selected.

Note If you select an existing column, its contents will be overwritten by the cell properties of custom members.

New column name

Type the name of the column that will store the cell properties for custom members. This option is available only if you select **Create a new column**.

A dimension table can have multiple columns (one per level) that store cell properties for custom members. Therefore, it is recommended that the new column name identify the column that stores the members to which the custom member options apply. For example, if the members are stored in the **store_country** column, type:

```
store_country_custom_member_options
```

Existing column

Select the column that will store the cell properties for custom members. This option is available only if you select **Use an existing column**.

OK

Click to temporarily save your changes and close the dialog box. To

permanently save your changes, after this dialog box closes, on the **File** menu, click **Save**.

Define Unary Operator Column Dialog Box

Use this dialog box to:

- Enable or disable the use of custom rollup operators.

- Create or select a column to store custom rollup operators. This column is created in or selected from the same dimension table that stores the members to which the custom rollup operators apply.

Custom rollup operators are simple mathematical functions that determine the cell values associated with members, and override the aggregate functions of measures. For write-enabled dimensions, custom rollup operators are created in the member properties pane of Dimension Editor (if the dimension is shared) or Dimension Browser. (This action requires that the write-enabled dimension is included in a cube that was processed since the dimension last changed.) For other dimensions, custom rollup operators are inserted into the dimension table using a tool outside Analysis Manager.

For more information about custom member formulas, see Custom Rollup Operators.

For related procedures and information on the behavior of specific custom rollup operators, see Using Custom Rollup Operators.

The **Define Unary Operator Column** dialog box appears when, in the tree pane of Dimension Editor or Cube Editor, you select a level, and in the **Advanced** tab of the properties pane, click the value beside **Unary Operators**, and then click the edit (**...**) button.

Options

Enable Unary Operators

Select to enable the use of custom rollup operators. This action sets the level's **Unary Operators** property to **True**. Cell values for members with custom rollup operators will be calculated according to the custom rollup operators.

Clear to disable the use of custom rollup operators. This action sets the level's **Unary Operators** property to **False**. Custom rollup operators will be ignored; cell values for members will be calculated according to the measures' aggregate functions.

Create a new column

Click to create a new column in which to store custom rollup operators. Specify the column name in the **New column name** box, which appears only when **Create a new column** is selected.

Use an existing column

Click to select an existing column in which to store custom rollup operators. Select the column in the **Existing column** box, which appears only when **Use an existing column** is selected.

Note If you select an existing column, its contents will be overwritten by the custom rollup operators.

New column name

Type the name of the column that will store custom rollup operators. This option is available only if you select **Create a new column**.

A dimension table can have multiple columns (one per level) that store custom rollup operators. Therefore, it is recommended that the new column name identify the column that stores the members to which the custom rollup operators apply. For example, if the members are stored in the **store_country** column, type:

```
store_country_custom_rollup_operator
```

Existing column

Select the column that will store custom rollup operators. This option is available only if you select **Use an existing column**.

OK

Click to temporarily save your changes and close the dialog box. To

permanently save your changes, after this dialog box closes, on the **File** menu, click **Save**.

Delete Member Dialog Box

Use this dialog box to:

- Confirm the deletion of a member.
- Select the disposition of its descendants.

CAUTION Clicking **OK** deletes the selected member. If you select **Delete its descendants**, clicking **OK** also deletes the member's descendants. Rows for these members are deleted from the dimension table. These deletions cannot be undone after you save dimension writeback changes.

This dialog box appears when you delete a member in the dimension members pane of Dimension Editor or Dimension Browser.

The following options are available only if the selected member has descendants.

Options

Delete its descendants

Select to delete the descendants of the deleted member.

Promote its descendants

Select to promote the descendants of the deleted member one level, to retain them as descendants of the deleted member's parent.

Dependency Network Browser

Use this tool to view the dependencies or relationships among objects in a data mining model.

Dependency Network Browser is displayed when, in the Analysis Manager tree pane, you right-click a data mining model and then click **Browse Dependency Network**.

In Dependency Network Browser, a data mining model is expressed as a network of attributes. Within the model, you can identify data dependencies and predictability among the related attributes. Dependency is indicated by arrows. The direction of predictability is indicated by arrowheads and by the color-coding of the nodes. For more information about why you should use this browser, see Viewing with Dependency Network Browser.

You can navigate the browser using keyboard shortcuts. For more information, see Keyboard Shortcuts (Dependency Network Browser).

Dependency Network Browser has four areas. For more information about the areas, click a link in the following table.

To do this	See
View a data mining model as a network of related attributes.	Network Browser Pane (Dependency Network Browser)
Change the view of the network browser pane and search for specific attributes by name.	Toolbar (Dependency Network Browser)
Use a slider to show the degree of dependency.	Slider Pane (Dependency Network Browser)
View and modify the color codes that indicate attribute state and predictability.	Legend Pane (Dependency Network Browser)

See Also

Building and Using Data Mining Models

Network Browser Pane (Dependency Network Browser)

Use the network browser pane to view the contents of a data mining model. In this pane, you can use toolbar tools to adjust the view to see more of the model or narrow the focus to see detail about specific areas.

When you first open a data mining model in the browser, attributes are positioned for you. However, you can move attributes by dragging them to new locations.

When you select an attribute, the color of its container changes to reflect selection state. In addition, the color scheme of surrounding nodes changes to reflect the direction of predictability. For example, for each attribute you select, the colors of the surrounding nodes indicate whether they predict the selected attribute, are predicted by the selected attribute, or both. Selecting an attribute also displays the legend pane, which provides a key to the color scheme associated with directions of predictability.

In the current view, you can reposition the slider bar to indicate the strength of dependency. As you move the slider up, the browser adds arrows corresponding to weaker dependencies.

You can search for specific attributes by name or add missing attributes to the network browser pane. If the model is very large, the initial presentation may omit attributes that are weakly related. To add these attributes to the network browser pane, click **Find** on the toolbar to select the node of interest. Adding an attribute also adds any additional nodes that are connected to it.

See Also

Viewing with Dependency Network Browser

Toolbar (Dependency Network Browser)

Slider Pane (Dependency Network Browser)

Legend Pane (Dependency Network Browser)

Keyboard Shortcuts (Dependency Network Browser)

Toolbar (Dependency Network Browser)

Use the toolbar to change the view of the network browser pane and select an attribute by name.

Icon	Label	Description
	Zoom In	Zoom in to the data mining model content in the network browser pane. With each successive click, the data mining model shrinks incrementally.
	Zoom Out	Zoom out of the data mining model content in the network browser pane. With each successive click, the data mining model expands incrementally.
	Zoom To Fit	Click to size the data mining model to the area of the network browser pane.
	Improve Layout	Click to optimize node layout so that the strongest relationships are visible.
	Find	Click to display the **Find Node** dialog box, where you can select an attribute by name.

See Also

Viewing with Dependency Network Browser

Dependency Network Browser

Slider Pane (Dependency Network Browser)

Use the slider pane to identify the strength of the relationship between attributes. The slider pane is a continuum between two points. As you move the slider along the continuum, the number of connections varies:

- **All links** shows the connections of all attributes that are visible in the network browser pane. Positioning the slider at **All links** shows how all attributes are related.

- **Strongest links** shows relationships among only those attributes that strongly predict the presence of the selected attribute.

See Also

Viewing with Dependency Network Browser

Dependency Network Browser

Legend Pane (Dependency Network Browser)

Use the legend pane to view the color codes that indicate attribute state and predictability. To change a color code, double-click the color swatch to open a color picker dialog box.

Icon	Label	Description
🟢	**Selected Node**	The attribute currently selected. You can only select one attribute at a time.
🔵	**Node that predicts it**	Indicates one or more attributes that predict the selected attribute.
🔴	**Node it predicts**	Indicates one or more attributes that the selected attribute predicts.
🟣	**Predicts both ways**	Between two related attributes, each attribute predicts and is predicted by the other.

See Also

Viewing with Dependency Network Browser

Dependency Network Browser

Keyboard Shortcuts (Dependency Network Browser)

This table describes the keyboard shortcuts available in Dependency Network Browser. These shortcuts correspond to toolbar buttons. For more information about the tools, see Toolbar (Dependency Network Browser).

Toolbar button	Shortcut
Zoom In	CTRL- +
Zoom Out	CTRL- -
Zoom To Fit	ALT- F
Improve Layout	ALT- L
Find	CTRL- F

See Also

Viewing with Dependency Network Browser

Dependency Network Browser

Dimension Browser

Use this tool to examine a dimension, including its members, member properties, member property values, and custom member formulas. Dimension Browser also permits the update of write-enabled dimensions.

Dimension Browser is displayed when you perform one of the following actions:

- In the Analysis Manager tree pane, right-click a shared dimension or virtual dimension, and then click **Browse Dimension Data**.
- In the Cube Editor tree pane, right-click a dimension, and then click **Browse**.

Dimension Browser displays the dimension in a graphical format, allowing you to view a multilevel dimension hierarchy.

Dimension Browser has three areas. For more information about the areas, click a link in the following table.

To do this	See
Display the dimension members in a tree format. Expand individual members to browse their descendants. For write-enabled dimensions, update the member hierarchy and the member names in the dimension members pane. Add, delete, and move members.	Dimension Members Pane (Dimension Browser)
View a list of the member properties of the member selected in the dimension members pane. For write-enabled dimensions, you	Member Properties Pane (Dimension Browser)

can update the member property values.	
View the custom member formula for the member selected in the dimension members pane. In some cases, create and update custom member formulas for that level's members.	Custom Member Formula Pane (Dimension Browser)

Dimension Members Pane (Dimension Browser)

Use this pane to display the dimension members in a tree format. Expand individual members to browse their descendants.

If the dimension is write-enabled, and included in a processed cube, you can update the member hierarchy and the member names in the dimension members pane. You can add, delete, and move members.

IMPORTANT Changes to a write-enabled dimension are recorded in the dimension table after you click **Close** and then click **Yes** to confirm the changes. Thereafter, these changes cannot be undone except by browsing the dimension again and manually reversing the changes.

To move a member, drag it to its new parent. To add, delete, move, or rename a member, right-click it, and then click one of the following shortcut menu options.

Shortcut menu option	Click to
Delete	Display the **Delete Member** dialog box, where you can confirm the deletion of the selected member and select the disposition of its descendants.
Move Down	Move the selected member (and its descendants) downward on its current level. The selected member becomes a child of its parent's following sibling.
Move Up	Move the selected member (and its descendants) upward on its current level. The selected member becomes a child of its parent's preceding sibling.
Indent	Move the selected member (and its descendants) down one level. The selected member's preceding sibling becomes its new parent. If the selected member does not have a preceding sibling, its following sibling becomes its new parent.
Outdent	Move the selected member (and its descendants)

	up one level. The selected member's grandparent becomes its new parent.
Rename	Rename the selected member.
New Member - Sibling	Display the **Create Member** dialog box, where you can add a sibling of the selected member.
New Member - Child	Display the **Create Member** dialog box, where you can add a child of the selected member.

See Also

Dimension Browser

Member Properties Pane (Dimension Browser)

Use this pane to view a list of the member properties of the member selected in the dimension members pane. The list of member properties contains all user-defined member properties for the selected member.

For each member property, the name is shown in the first column of the pane, and the value is shown in the second column.

If the dimension is write-enabled and is included in a processed cube, you can use the member properties pane to update the member property values.

Important Changes to member property values are recorded in the dimension table after you click **Close** and then click **Yes** to confirm the changes. Thereafter, these changes cannot be undone except by browsing the dimension again and manually reversing the changes.

See Also

Dimension Browser

Custom Member Formula Pane (Dimension Browser)

Use this pane to create and update custom member formulas for members of the selected level, if certain conditions are met. The browsed dimension must be write-enabled, and one of its level's **Custom Members** properties must be set to **True**.

The custom member formula pane displays the custom member formula for the member selected in the dimension members pane. If no formula is visible, none is defined.

A custom member formula is a Multidimensional Expressions (MDX) expression that determines the cell values associated with a member and overrides the aggregate functions of measures. For more information about custom member formulas, see Custom Rollup Formulas and Custom Member Formulas.

The custom member formulas are stored in the dimension table that contains the members.

Important Changes to custom member formulas are recorded in the dimension table after you click **Close** and then click **Yes** to confirm the changes. Thereafter, these changes cannot be undone except by browsing the dimension again and manually reversing the changes.

Options

Dimension Members

View dimension members in a tree format. Expand individual members to browse their descendents. Select a member to create or update members of that level.

Member properties

View member properties for the selected member. You can use this pane to modify properties if the dimension is write-enabled.

Custom Member Formula

View the custom member formula for the member selected in the dimension members pane. If no formula is visible, none is defined. You can use the custom member formulas pane to create and update custom member formulas for members of that level, if certain conditions are met. The browsed dimension must be write-enabled, and one of its level's **Custom Members** properties must be set to **True**.

Arithmetic operator buttons

Click to add operators to a formula.

Edit (...) button

Click to open MDX Builder, where you can create an MDX expression.

For more information, see Creating Custom Member Formulas.

See Also

Custom Rollup Formulas and Custom Member Formulas

Dimension Browser

Dimension Editor - Data View

Use this tool to browse a shared dimension's members, member properties, and custom member formulas. You can also examine and edit the structure of all types of shared dimensions, including parent-child and virtual dimensions, and their levels. You can also use Dimension Editor and connected dialog boxes to perform various procedures with shared dimensions.

Dimension Editor appears in two views, data and schema. Both views include the tree pane and properties pane. You can switch from one view to another by clicking the **Data** tab or the **Schema** tab at the bottom of Dimension Editor, or by clicking **Data** or **Schema** on the **View** menu.

This topic describes the data view. For information about the schema view, see Dimension Editor - Schema View. The data view is shown here.

Dimension Editor appears when you do either of the following:

- In the Analysis Manager tree pane, under a database, open the Shared Dimensions folder. Right-click a dimension, and then click **Edit**.

- In the Analysis Manager tree pane, under a database, open the Shared Dimensions folder. Click a dimension, and then on the **Action** menu, click **Edit**.

Dimension Editor has five areas. For more information about the areas, click a link in the following table.

To do this	See
Perform commands available in Dimension Editor menus.	Menus (Dimension Editor Data View)
Perform common actions represented by icons on the Dimension Editor toolbar.	Toolbar (Dimension Editor Data View)
Display the objects in the dimension.	Tree Pane (Dimension Editor Data View)

Access associated dialog boxes.	
Display the properties of the object selected in the tree pane. Modify the property settings for the selected object.	Properties Pane (Dimension Editor Data View)
Browse members of a shared dimension, member properties, and custom member formulas.	Data Tab (Dimension Editor Data View)

See Also

Creating Custom Member Formulas

Creating a Shared Dimension with the Editor

Creating Member Groups

Creating Member Properties

Menus (Dimension Editor Data View)

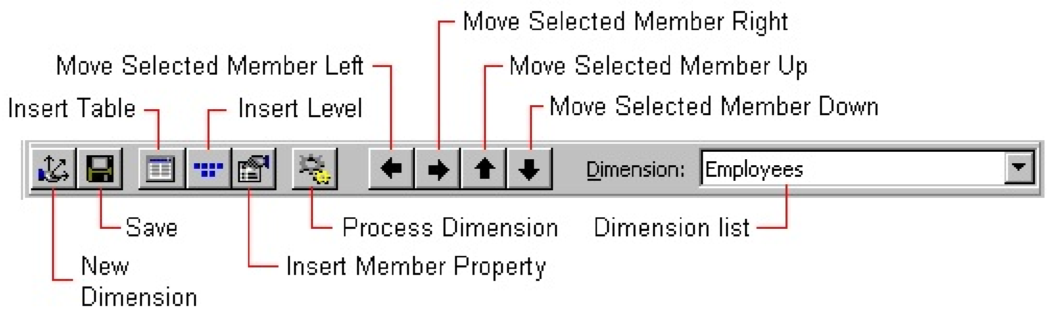
The following options are available through menus in Dimension Editor.

Menu	Option	Description
File	**New Dimension - Wizard**	Starts the Dimension Wizard so you can create a new shared dimension.
	New Dimension - Editor	Displays the **Choose a Dimension Table** dialog box so you can begin creating a shared dimension with the editor.
	Save	Saves the dimension. If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
	Save As	Saves the dimension under a different name.
	Exit	Closes Dimension Editor.
Edit	**Rename**	Renames the selected object.
	Delete	Deletes the selected object.
	Remove Join	Removes all joins from the selected column (schema view only).
View	**Schema**	Displays the schema view, including the **Schema** tab.
	Data	Displays the data view, including the **Data** tab.
	Properties	Expands or collapses the properties pane.
Insert	**Tables**	Displays the **Select Table** dialog box so you can add tables to the dimension.
	Level	Displays the **Insert Level** dialog box so you can add a new level.

	Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property in the selected level.
	Join	Displays the **Join Columns** dialog box, where you can add a join to the selected column (schema view only).
Tools	**Process Dimension**	Processes the dimension.
	Count Dimension Members	Counts the number of members in a dimension.
	Validate Dimension Structure	Verifies that the dimension structure does not contain invalid components that would prevent processing of the dimension.
Help	**Help on Dimension Editor**	Displays a Help topic about Dimension Editor.
	Contents and Index	Opens SQL Server Books Online.

Toolbar (Dimension Editor Data View)

Use the following toolbar buttons to perform common operations.



Button	Description
New Dimension	Starts the Dimension Wizard so you can create a new shared dimension.
Save	Saves the dimension.If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
Insert Table	Displays the **Select table** dialog box where you can add tables to the dimension. This button is not available for virtual dimensions.
Insert Level	Displays the **Insert Level** dialog box so you can add a new level.
Insert Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property in the selected level.
Process Dimension	Displays the **Process a Dimension** dialog box so you can incrementally update or rebuild the selected dimension.
Move Selected Member Left*	In the dimension members pane, moves the selected member (and its descendants) up one level. The selected member's grandparent becomes its new parent.
Move Selected Member Right*	In the dimension members pane, moves the selected member (and its descendants) down one level. The

	selected member's preceding sibling becomes its new parent.
Move Selected Member Up*	In the dimension members pane, moves the selected member (and its descendants) upward on its current level. The preceding sibling of the selected member's parent becomes its new parent.
Move Selected Member Down*	In the dimension members pane, moves the selected member (and its descendants) downward on its current level. The following sibling of the selected member's parent becomes its new parent.

* This button is visible only when you are in data view and editing a write-enabled dimension that is included in a cube that was processed since the dimension last changed.

Tree Pane (Dimension Editor Data View)

Use the tree pane to display the objects in the dimension. Right-click an object to see a shortcut menu for that object.

Right-click	Shortcut menu option	Description
Dimension	**New Dimension**	Displays the **Choose a Dimension Table** dialog box, where you can begin creating a shared dimension with the editor.
	Process Dimension	Processes the dimension.
	New Level	Displays the **Insert Level** dialog box, where you can add a new level.
Level	**New Level**	Displays the **Insert Level** dialog box, where you can add a new level.
	New Member Property	Displays the **Insert Member Property** dialog box, where you can add a new member property in the level.
	Delete	Deletes the level.
	Rename	Renames the level.
Member Property Folder	**New Member Property**	Displays the **Insert Member Property** dialog box, where you can add a new member property.
Member Property	**New Member Property**	Displays the **Insert Member Property** dialog box, where you can add a new member property.
	Delete	Deletes the member property.
	Rename	Renames the member property.

Properties Pane (Dimension Editor Data View)

Use the properties pane to display the properties of the object selected in the tree pane. To display the properties pane, click the **Properties** button.

Each type of object (dimension, level, or member property) contains a different set of properties. Use the properties pane to modify the property settings for the selected object.

The following table describes the properties displayed in the properties pane.

Object	Property	Description
Dimension	**All Caption**	The name of the member in the (All) level.
	All Level	Indicates whether the dimension contains an (All) level. If the value is **Yes**, the (All) level is the top level of the dimension but is not displayed in the Dimension Editor tree pane. The (All) level contains a single member whose cell value is the aggregate of cell values for all members in the next lower level.
	All Member Formula	Stores the MDX expression used to override the default rollup of the All member.
	Allow Duplicate Names	Indicates whether the members under a common parent can have the same name.
	Changing	Indicates whether the dimension is optimized for frequent changes. If the value is **True**, query performance may be slower. However, levels and members below the top level and above the bottom level can be added, moved, and deleted, and the subsequent processing requirement is eliminated or reduced. Consequently, interruptions of end users' access to the cubes that include the dimension

		can be reduced. For more information, see Changing Dimensions.
	Data Member Caption template	Controls the names of data members when the dimension's **Members With Data** property is set to **Nonleaf data visible**. Type a value that includes an asterisk (*). The name of each data member will be the value with the asterisk replaced by the parent member's name. The **Data Member Caption template** property is available only for parent-child dimensions.
	Data source	Indicates the data source that contains the dimension table(s).
	Default Member	Indicates the member that slices the datasets returned by queries when the dimension is not displayed on an axis and no slicing member in the dimension is specified. If no default member is specified, and the dimension's **All Level** property is **Yes**, the member indicated by the **All Caption** property is the default member. If no default member is specified, and the dimension's **All Level** property is **No**, an arbitrary member of the highest level is the default member.
	Depends on Dimension	For virtual dimensions, indicates the dimension that supplies the member properties or columns on which the virtual dimension's levels are based. For dimensions that are not virtual, this is the dimension according to which aggregation design is optimized. A dimension in this property's value is advantageous when the cross product of the two dimensions' members results in a significant percentage of combinations that cannot coexist. For example, a Customer Gender dimension's

		Depends on Dimension property is **Customers**. Fifty percent of the combinations resulting from the cross product of the dimensions' lowest-level members cannot coexist because a customer can have only one gender. For more information, see Dependent Dimensions.
	Description	Contains the description of the dimension.
	Enable Real-Time Updates	Indicates whether or not the dimension supports real-time updates. For this to be set to **True**, the dimension must use a ROLAP partition and a Microsoft® SQL Server™ 2000 data source. For more information, see Real-Time Cubes.
	Member Keys Unique	Indicates whether member keys are unique within the dimension. If the value of this property is changed, process the dimension using the **Rebuild the dimension structure** option. This is read-only and set to **True** for changing dimensions. For more information, see Updating and Rebuilding Shared Dimensions.
	Member Names Unique	Indicates whether member names are unique within the dimension. If the value is **True**, internal member names omit qualifying level names and member names. These internal member names are used in Multidimensional Expressions (MDX) expressions.
	Members with Data	Indicates whether nonleaf members are allowed to have associated fact table data. If they are allowed, this property also indicates whether children of nonleaf members are created to display this data. Valid values are: • **Leaf members only**: Only leaf members can have associated fact

		table data. If a nonleaf member has associated fact table data, processing fails. This value is the default. • **Nonleaf data hidden**: Nonleaf members can have associated fact table data. This data is not represented among the nonleaf members' descendants. Consequently, it might appear to end users that values aggregate incorrectly. • **Nonleaf data visible**: Nonleaf members can have associated fact table data. This data is represented among the nonleaf members' descendants by the creation of a child for each nonleaf member. This child, called a data member, is a leaf member and has a value equal to the aggregate of its parent's associated fact table data. The data members' names are controlled by the **Data Member Caption Template** property. This property is available only for parent-child dimensions.
	Name	Indicates the name of the dimension.
	Source Table Filter	Lists the WHERE clause expression that is applied to the dimension table to limit the members in the dimension. For example, in the Store dimension supplied with SQL Server 2000 Analysis Services, to include

		only the Canada member and its descendants, type: "store"."store_country" = 'Canada'
	Storage Mode	Determines the type of storage for the dimension. If the value is **MOLAP** (multidimensional OLAP), the dimension data is stored in a multidimensional structure on the Analysis server. If the value is **ROLAP** (relational OLAP), the dimension data is the dimension table itself. **MOLAP** provides better performance and is recommended except for extremely large dimensions (that is, dimensions that have approximately 5 to 10 million members). In order to select **ROLAP**, the lowest level's **Member Keys Unique** property must be **True**. Restrictions also apply to the dimension's **Aggregation Usage** property in all cubes in which the dimension is used. For more information, see Dimension Storage Modes.
	Type	Indicates the type of the dimension. Standard is the default. This property indicates to client applications the kind of information in the dimension.
	Virtual	Indicates whether the dimension is a virtual dimension. If you set this property to **True** while the **Depends on Dimension** property is (None), the tables in the **Schema** tab disappear. In the **Depends on Dimension** property, select the dimension that supplies the columns or member properties on which the edited dimension is based. The tables for the selected dimension appear in the **Schema** tab.

	Write-Enabled	Indicates whether the dimension's members can be updated while administrators browse the dimension and while end users browse cubes that contain the dimension. The only end users that can update a write-enabled dimension are those in cube roles granted read/write access to the dimension. Only parent-child dimensions can be write-enabled. For more information, see Write-Enabled Dimensions.
Level	**Custom Members**	Indicates whether custom member formulas are used to determine members' cell values. For more information, see Custom Rollup Formulas and Custom Member Formulas. To view the custom member formulas, on the **View** menu click **Data**, and then see the custom member formula pane.
	Custom Member Options	Indicates whether calculation options can be defined for custom members on this level and unary operators on the following level. Calculation options are stored in a column in the dimension table. To set this property to **True**, you must first set the **Custom Members** property to **True**.
	Custom Rollup Formula	Stores the MDX expression used to override the default rollup of values in the level.
	Description	Contains the description of the dimension.
	Grouping	Indicates whether the level contains member groups. Member groups can be used to satisfy the maximum limit of 64,000 members under a parent. To use member groups in this way, create a new level immediately above and identical to the level that exceeds the limit, and then set the new level's **Grouping** property to **Automatic**. For more

		information, see Member Groups.
	Hide Member If	Determines which members are hidden from end users as they browse cubes. Hidden members support ragged dimensions, which contain logical gaps in member lineage, by hiding the members that occupy the gaps. Valid values are: • **Never hidden**: No members are hidden. • **No name**: Every member whose name is null or an empty string is hidden. • **Parent's name**: Every member with the same name as its parent is hidden. • **Only child with no name**: Every member that is an only child and whose name is null or an empty string is hidden. • **Only child with parent's name**: Every member that is an only child and has the same name as its parent is hidden. This property is not available for parent-child dimensions.
	Key Data Size	Indicates the size (in bytes) of the columns that store member keys in aggregations. Member keys are copied from the column

		specified in the **Member Key Column** property.
	Key Data Type	Indicates the data type of the columns that store member keys in aggregations. Member keys are copied from the column specified in the **Member Key Column** property.
	Level Naming Template	Determines the level names displayed to end users when they browse cubes containing the dimension. This property is available only for parent-child dimensions. Click this property and click the edit (**...**) button to display the **Level Naming Template** dialog box.
	Level Type	Indicates the type of the level. **Regular** is the default. The following values are used only in dimensions whose **Type** property is **Time**: Years, Half-Years, Quarters, Months, Weeks, Days, Hours, Minutes, Seconds, and Time-Undefined. The **Level Type** property indicates to client applications the kind of information in the level. This property is not displayed for parent-child dimensions.
	Member Count	Indicates the number of members in the level at the time they were last counted by Analysis Services, or a user-provided estimate of the member count. You can update this value by clicking **Count Dimension Members** on the **Tools** menu.
	Member Key Column	Contains the member keys. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Keys	Indicates whether member keys are unique

	Unique	within the level. This property is not available for parent-child dimensions. If the value of this property is changed, process the level's corresponding dimension using the **Rebuild the dimension structure** option. For more information, see Updating and Rebuilding Shared Dimensions.
	Member Name Column	Indicates the column that contains the member names, which are displayed to end users as they browse cubes. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Names Unique	Indicates whether member names are unique within the level. If the value is **True**, internal member names omit qualifying member names. These internal member names are used in MDX expressions.
	Name	Contains the name of the level.
	Order By	Determines the sort order for displayed members. You can sort by member name, member key, or any member property defined for the level.
	Parent Key Column	Indicates the column that contains the keys for the members' parents. This property is available only for parent-child dimensions.
	Root Member If	Indicates the criteria by which members of the highest level (ignoring the (All) level, if any) are identified. The following values are valid:

		• **Parent is blank, self, or missing**: A member is in the highest level if any of the following three criteria is met. • **Parent is blank**: A member is in the highest level if its value in the **Parent Key Column** is null or 0 (zero). • **Parent is self**: A member is in the highest level if its value in the **Parent Key Column** is equal to its value in the **Member Key Column**. • **Parent is missing**: A member is in the highest level if its value in the **Parent Key Column** does not exist in the **Member Key Column**. This property is available only for parent-child dimensions.
	Skipped Levels Column	Indicates the column that contains the number of levels between a member and its parent, excluding the member and parent. Valid values in the column are 0 (zero) and positive integers. This property is available only for parent-child dimensions.
	Unary Operators	Enables unary operators (custom rollup operators) that control how level members are aggregated into their parent member's value. For more information, see Custom Rollup Operators. To view the custom rollup operators, on the **View** menu click **Data**, and then see the value beside UNARY_COLUMN in the member

		properties pane.
	Visible	Indicates whether the level is visible to end users as they browse cubes. To set this property to **False**, you must first set the **Member Keys Unique** property of all lower levels to **True**. The **Visible** property is not available for parent-child dimensions. This property's value can be overridden for individual cubes by the **Visible** property in Cube Editor.
Member property	**Caption**	Contains the caption used to display the member property.
	Data Size	Indicates the maximum number of characters allowed in the column that stores the member property values.
	Data Type	Indicates the data type of the column that stores the member property values.
	Description	Contains a description of the member property.
	Language	Indicates the language used to display the member property.
	Name	Indicates the name of the member property.
	Source Column	Indicates the column that stores the values of the member property. This column must be in one of the dimension tables for the dimension that contains the member property.
	Type	Provides an indicator to client applications of the type of information in the member property values.
	Visible	Indicates whether the member property is visible to end users as they browse the cube.

Data Tab (Dimension Editor Data View)

Use this tab to browse a shared dimension's members, member properties, and custom member formulas. The following topics describe panes in the **Data** tab.

- Dimension Members Pane (Dimension Editor Data View)
- Member Properties Pane (Dimension Editor Data View)
- Custom Member Formula Pane (Dimension Editor Data View)

Dimension Members Pane (Dimension Editor Data View)

Displays the dimension members in a tree format. Expand individual members to browse their descendants.

If the dimension is write-enabled and included in a processed cube, you can update the member hierarchy and member names in the dimension members pane. You can add, delete, and move members.

Caution Changes to a write-enabled dimension are recorded in the dimension table when you save the dimension. Thereafter, these changes cannot be undone except by editing the dimension and manually reversing the changes.

To move a member, drag it to its new parent or use the **Move Selected Member** buttons on the toolbar. To add, delete, move, or rename a member, right-click the member, and then click one of the following shortcut menu options.

Shortcut menu option	Description
Delete	Displays the **Delete Member** dialog box so you can confirm the deletion of the selected member and select the disposition of its descendants.
Move Down	Moves the selected member (and its descendants) downward on its current level. The selected member becomes a child of its parent's following sibling.
Move Up	Moves the selected member (and its descendants) upward on its current level. The selected member becomes a child of its parent's preceding sibling.
Indent	Moves the selected member (and its descendants) down one level. The selected member's preceding sibling becomes its new parent. If the selected member does not have a preceding sibling, its following sibling becomes its new parent.
Outdent	Moves the selected member (and its descendants)

	up one level. The selected member's grandparent becomes its new parent.
Rename	Renames the selected member.
New Member - Sibling	Displays the **Create Member** dialog box so you can add a sibling of the selected member.
New Member - Child	Displays the **Create Member** dialog box so you can add a child of the selected member.

Member Properties Pane (Dimension Editor Data View)

Lists the member properties of the member selected in the dimension members pane. The list of member properties contains all user-defined member properties for the selected member.

For each member property, the name is shown in the first column of the pane and the value is shown in the second column. The member properties are sorted in the same order as in the tree pane.

If the dimension is write-enabled and is included in a processed cube, you can use the member properties pane to update the member property values.

CAUTION Changes to member property values are recorded in the dimension table when you save the dimension. Thereafter, these changes cannot be undone except by editing the dimension and manually reversing the changes.

Custom Member Formula Pane (Dimension Editor Data View)

Contains a custom member formula, which is an expression in Multidimensional Expressions (MDX) that determines the cell values associated with a member and overrides the aggregate functions of measures. For general information about custom member formulas, see Custom Rollup Formulas and Custom Member Formulas.

The custom member formula pane displays the custom member formula for the member selected in the dimension members pane. If no custom member formula is visible, none is defined.

You can use the custom member formulas pane to create and update custom member formulas for members of that level, if certain conditions are met. The browsed dimension must be write-enabled, it must be included in a processed cube, and one of the **Custom Members** property for one of its levels must be set to **True**. You can use any combination of the following methods:

- Type.
- Click the arithmetic operator buttons and parentheses buttons.
- Click the edit button (**...**) to display MDX Builder, and then drag and drop items to construct the custom member formula.

The custom member formulas are stored in the dimension table that contains the members.

Caution Changes to custom member formulas are recorded in the dimension table when you save the dimension. Thereafter, these changes cannot be undone except by editing the dimension and manually reversing the changes.

For more information, see Creating Custom Member Formulas.

Dimension Editor - Schema View

Use this tool to view and edit a shared dimension's schema. You can also examine and edit the structure of all types of shared dimensions, including parent-child and virtual dimensions, and their levels. You can also use Dimension Editor and connected dialog boxes to perform various procedures with shared dimensions.

Dimension Editor appears in two views, data and schema. Both views include the tree pane and properties pane. You can switch from one view to another by clicking the **Data** tab or the **Schema** tab at the bottom of Dimension Editor, or by clicking **Data** or **Schema** on the **View** menu.

This topic describes the schema view. For information about the data view, see Dimension Editor - Data View. The schema view is shown here.

Dimension Editor appears when you do either of the following:

- In the Analysis Manager tree pane, under a database, open the Shared Dimensions folder. Right-click a dimension, and then click **Edit**.

- In the Analysis Manager tree pane, under a database, open the Shared Dimensions folder. Click a dimension, and then on the **Action** menu, click **Edit**.

- Dimension Editor has four areas. For more information about the areas, click a link in the following table.

To do this	**See**
Perform commands available in Dimension Editor menus.	Menus
Perform common actions represented by icons on the Dimension Editor toolbar.	Toolbar
Display objects in the dimension.	Tree Pane

Access associated dialog boxes and available information about dimension objects.	
Display the properties of the object selected in the tree pane. Modify the property settings for the selected object.	Properties Pane
Display the dimension tables. Add tables and levels, and browse the data in the dimension tables. View columns in a table.	Schema Tab

See Also

Creating a Shared Dimension with the Editor

Creating Custom Member Formulas

Creating Member Groups

Creating Member Properties

Menus (Dimension Editor Schema View)

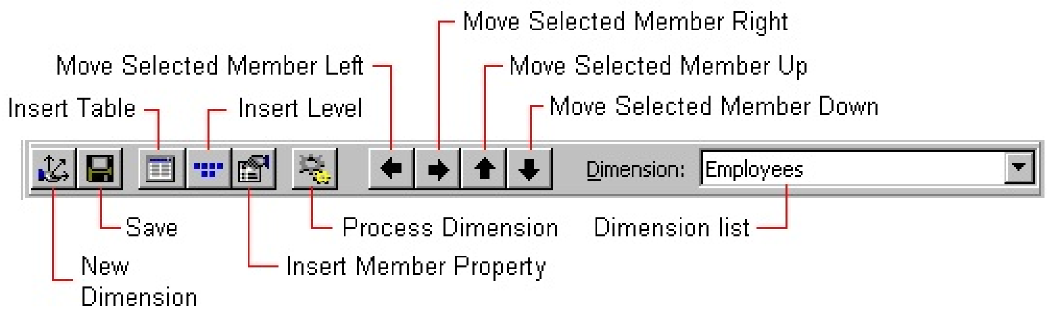
The following options are available through menus in Dimension Editor.

Menu	Option	Description
File	**New Dimension - Wizard**	Starts the Dimension Wizard so you can create a new shared dimension.
	New Dimension - Editor	Displays the **Choose a Dimension Table** dialog box so you can begin creating a shared dimension with the editor.
	Save	Saves the dimension. If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
	Save As	Saves the dimension under a different name.
	Exit	Closes Dimension Editor.
Edit	**Rename**	Renames the selected object.
	Delete	Deletes the selected object.
	Remove Join	Removes all joins from the selected column (schema view only).
View	**Schema**	Displays the schema view, including the **Schema** tab.
	Data	Displays the data view, including the **Data** tab.
	Properties	Expands or collapses the properties pane.
Insert	**Tables**	Displays the **Select Table** dialog box so you can add tables to the dimension.
	Level	Displays the **Insert Level** dialog box so you can add a new level.

	Member Property	Displays the **Insert Member Property** dialog box so you can add a new member property in the selected level.
	Join	Displays the **Join Columns** dialog box, where you can add a join to the selected column (schema view only).
Tools	**Process Dimension**	Processes the dimension.
	Count Dimension Members	Counts the number of members in a dimension.
	Validate Dimension Structure	Verifies that the dimension structure does not contain invalid components that would prevent processing of the dimension.
Help	**Help on Dimension Editor**	Displays a Help topic about Dimension Editor.
	Contents and Index	Opens SQL Server Books Online.

Toolbar (Dimension Editor Schema View)

Use the following toolbar buttons to perform common operations.



Button	**Description**
New Dimension	Starts the Dimension Wizard, where you can create a new shared dimension.
Save	Saves the dimension.If you are saving a parent-child or changing dimension, you will be prompted to perform an incremental processing of the dimension.
Insert Table	Displays the **Select table** dialog box so you can add tables to the dimension. This button is not available for virtual dimensions.
Insert Level	Displays the **Insert Level** dialog box, where you can add a new level.
Insert Member Property	Displays the **Insert Member Property** dialog box, where you can add a new member property in the selected level.
Process Dimension	Displays the **Process a Dimension** dialog box, where you can incrementally update or rebuild the selected dimension.
Move Selected Member Left*	In the dimension members pane, moves the selected member (and its descendants) up one level. The selected member's grandparent becomes its new parent.
Move Selected Member Right*	In the dimension members pane, moves the selected member (and its descendants) down one level. The

	selected member's preceding sibling becomes its new parent.
Move Selected Member Up*	In the dimension members pane, moves the selected member (and its descendants) upward on its current level. The preceding sibling of the selected member's parent becomes its new parent.
Move Selected Member Down*	In the dimension members pane, moves the selected member (and its descendants) downward on its current level. The following sibling of the selected member's parent becomes its new parent.

* This button is visible only when you are in data view and editing a write-enabled dimension that is included in a cube that was processed since the dimension last changed.

Tree Pane (Dimension Editor Schema View)

Use the tree pane to display the objects in the dimension. Right-click an object to see a shortcut menu for that object.

The following table lists how to access available information about dimension objects.

Right-click	Shortcut menu option	Description
Dimension	**New Dimension**	Displays the **Choose a Dimension Table** dialog box, where you can begin creating a shared dimension with the editor.
	Process Dimension	Processes the dimension.
	New Level	Displays the **Insert Level** dialog box, where you can add a new level.
Level	**New Level**	Displays the **Insert Level** dialog box, where you can add a new level.
	New Member Property	Displays the **Insert Level** dialog box, where you can add a new member property in the level.
	Delete	Deletes the level.
	Rename	Renames the level.
Member Property Folder	**New Member Property**	Displays the **Insert Member Property** dialog box, where you can add a new member property.
Member Property	**New Member Property**	Displays the **Insert member Property** dialog box, where you can add a new member property.
	Delete	Deletes the member property.
	Rename	Renames the member property.

Properties Pane (Dimension Editor Schema View)

Use the properties pane to display the properties of the object selected in the tree pane. To display the properties pane, click the **Properties** button.

Each type of object (dimension, level, or member property) contains a different set of properties. Use the properties pane to modify the property settings for the selected object.

The following table describes the properties displayed in the properties pane.

Object	Property	Description
Dimension	**All Caption**	Indicates the name of the member in the (All) level.
	All Level	Indicates whether the dimension contains an (All) level. If the value is **Yes**, the (All) level is the top level of the dimension but is not displayed in the Dimension Editor tree pane. The (All) level contains a single member whose cell value is the aggregate of cell values for all members in the next lower level.
	Allow Duplicate Names	Indicates whether the members under a common parent can have the same name.
	All Member Formula	Stores the MDX expression used to override the default rollup of the All member.
	Changing	Indicates whether the dimension is optimized for frequent changes. If the value is **True**, query performance may be slower. However, levels and members below the top level and above the bottom level can be added, moved, and deleted, and the subsequent processing requirement is eliminated or reduced. Consequently, interruptions of end users' access to the cubes that include the dimension

		can be reduced. For more information, see Changing Dimensions.
	Data Member Caption template	Controls the names of data members when the dimension's **Members With Data** property is set to **Nonleaf data visible**. Type a value that includes an asterisk (*). The name of each data member will be the value with the asterisk replaced by the parent member's name. The **Data Member Caption template** property is available only for parent-child dimensions.
	Data source	Indicates the data source that contains the dimension table(s).
	Default Member	Indicates the member that slices the datasets returned by queries when the dimension is not displayed on an axis and no slicing member in the dimension is specified. If no default member is specified, and the dimension's **All Level** property is **Yes**, the member indicated by the **All Caption** property is the default member. If no default member is specified, and the dimension's **All Level** property is **No**, an arbitrary member of the highest level is the default member.
	Depends on Dimension	For virtual dimensions, indicates the dimension that supplies the member properties or columns on which the virtual dimension's levels are based. For dimensions that are not virtual, this is the dimension according to which aggregation design is optimized. A dimension in this property's value is advantageous when the cross product of the two dimensions' members results in a significant percentage of combinations that cannot coexist. For example, a Customer Gender dimension's

		Depends on Dimension property is **Customers**. Fifty percent of the combinations resulting from the cross product of the dimensions' lowest-level members cannot coexist because a customer can have only one gender. For more information, see Dependent Dimensions.
	Description	Contains the description of the dimension.
	Enable Real-Time Updates	Indicates whether or not the dimension supports real-time updates. For this to be set to **True**, the dimension must use a ROLAP partition and a Microsoft® SQL Server™ 2000 data source. For more information, see Real-Time Cubes.
	Member Keys Unique	Indicates whether member keys are unique within the dimension. If the value of this property is changed, process the dimension using the **Rebuild the dimension structure** option. This is read-only and set to **True** for changing dimensions. For more information, see Updating and Rebuilding Shared Dimensions.
	Member Names Unique	Indicates whether member names are unique within the dimension. If the value is **True**, internal member names omit qualifying level names and member names. These internal member names are used in Multidimensional Expressions (MDX) expressions.
	Members with Data	Indicates whether nonleaf members are allowed to have associated fact table data. If they are allowed, this property also indicates whether children of nonleaf members are created to display this data. Valid values are: • **Leaf members only**: Only leaf members can have associated fact

		table data. If a nonleaf member has associated fact table data, processing fails. This value is the default. • **Nonleaf data hidden**: Nonleaf members can have associated fact table data. This data is not represented among the nonleaf members' descendants. Consequently, it might appear to end users that values aggregate incorrectly. • **Nonleaf data visible**: Nonleaf members can have associated fact table data. This data is represented among the nonleaf members' descendants by the creation of a child for each nonleaf member. This child, called a data member, is a leaf member and has a value equal to the aggregate of its parent's associated fact table data. The data members' names are controlled by the **Data Member Caption Template** property. This property is available only for parent-child dimensions.
	Name	Contains the name of the dimension.
	Source Table Filter	Indicates the WHERE clause expression applied to the dimension table to limit the members in the dimension. For example, in the Store dimension supplied with SQL Server 2000 Analysis Services, to include

		only the Canada member and its descendants, type: "store"."store_country" = 'Canada'
	Storage Mode	Indicates the type of storage for the dimension. If the value is **MOLAP** (multidimensional OLAP), the dimension data is stored in a multidimensional structure on the Analysis server. If the value is **ROLAP** (relational OLAP), the dimension data is the dimension table itself. **MOLAP** provides better performance and is recommended except for extremely large dimensions (that is, dimensions that have approximately 5 to 10 million members). In order to select **ROLAP**, the lowest level's **Member Keys Unique** property must be **True**. Restrictions also apply to the dimension's **Aggregation Usage** property in all cubes in which the dimension is used. For more information, see Dimension Storage Modes.
	Type	Indicates the type of the dimension. Standard is the default. This property indicates to client applications the kind of information in the dimension.
	Virtual	Indicates whether the dimension is a virtual dimension. If you set this property to **True** while the **Depends on Dimension** property is (None), the tables in the **Schema** tab disappear. In the **Depends on Dimension** property, select the dimension that supplies the columns or member properties on which the edited dimension is based. The tables for the selected dimension appear in the **Schema** tab.

	Write-Enabled	Indicates whether the dimension's members can be updated while administrators browse the dimension and while end users browse cubes that contain the dimension. The only end users that can update a write-enabled dimension are those in cube roles granted read/write access to the dimension. Only parent-child dimensions can be write-enabled. For more information, see Write-Enabled Dimensions.
Level	**Custom Members**	Indicates whether custom member formulas are used to determine members' cell values. For more information, see Custom Rollup Formulas and Custom Member Formulas. To view the custom member formulas, on the **View** menu click **Data**, and then see the custom member formula pane.
	Custom Member Options	Indicates whether calculation options can be defined for custom members on this level and unary operators on the following level. Calculation options are stored in a column in the dimension table. To set this property to **True**, you must first set the **Custom Members** property to **True**.
	Custom Rollup Formula	Stores the MDX expression used to override the default rollup of values in the level.
	Description	Contains a description of the level.
	Grouping	Indicates whether the level contains member groups. Member groups can be used to satisfy the maximum limit of 64,000 members under a parent. To use member groups in this way, create a new level immediately above and identical to the level that exceeds the limit, and then set the new level's **Grouping** property to **Automatic**. For more

		information, see Member Groups.
	Hide Member If	Determines which members are hidden from end users as they browse cubes. Hidden members support ragged dimensions, which contain logical gaps in member lineage, by hiding the members that occupy the gaps. Valid values are: • **Never hidden**: No members are hidden. • **No name**: Every member whose name is null or an empty string is hidden. • **Parent's name**: Every member with the same name as its parent is hidden. • **Only child with no name**: Every member that is an only child and whose name is null or an empty string is hidden. • **Only child with parent's name**: Every member that is an only child and has the same name as its parent is hidden. This property is not available for parent-child dimensions.
	Key Data Size	Indicates the size (in bytes) of the columns that store member keys in aggregations. Member keys are copied from the column

		specified in the **Member Key Column** property.
	Key Data Type	Indicates the data type of the columns that store member keys in aggregations. Member keys are copied from the column specified in the **Member Key Column** property.
	Level Naming Template	Determines the level names displayed to end users when they browse cubes containing the dimension. This property is available only for parent-child dimensions. Click this property and click the edit (**...**) button to display the **Level Naming Template** dialog box.
	Level Type	Indicates the type of the level. Regular is the default. The following values are used only in dimensions whose **Type** property is Time: Years, Half-Years, Quarters, Months, Weeks, Days, Hours, Minutes, Seconds, and Time-Undefined. The **Level Type** property indicates to client applications the kind of information in the level. This property is not displayed for parent-child dimensions.
	Member Count	Indicates the number of members in the level at the time they were last counted by Analysis Services, or a user-provided estimate of the member count. You can update this value by clicking **Count Dimension Members** on the **Tools** menu.
	Member Key Column	Indicates the column that contains the member keys. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.

	Member Keys Unique	Indicates whether member keys are unique within the level. This property is not available for parent-child dimensions. If the value of this property is changed, process the level's corresponding dimension using the **Rebuild the dimension structure** option. For more information, see Updating and Rebuilding Shared Dimensions.
	Member Name Column	Indicates the column that contains the member names, which are displayed to end users as they browse cubes. Alternatively, the value can be derived from a column, such as an expression that extracts the year value from a date-formatted column. The syntax of such expressions must comply with the requirements of the data source provider; otherwise, processing fails with one or more errors.
	Member Names Unique	Indicates whether member names are unique within the level. If the value is **True**, internal member names omit qualifying member names. These internal member names are used in MDX expressions.
	Name	Contains the name of the level.
	Order By	Determines the sort order for displayed members. You can sort by member name, member key, or any member property defined for the level.
	Parent Key Column	Indicates the column that contains the keys for the members' parents. This property is available only for parent-child dimensions.
	Root Member If	Indicates the criteria by which members of the highest level (ignoring the (All) level, if any) are identified. The following values are valid:

		<ul><li>**Parent is blank, self, or missing**: A member is in the highest level if any of the following three criteria is met.</li><li>**Parent is blank**: A member is in the highest level if its value in the **Parent Key Column** is null or 0 (zero).</li><li>**Parent is self**: A member is in the highest level if its value in the **Parent Key Column** is equal to its value in the **Member Key Column**.</li><li>**Parent is missing**: A member is in the highest level if its value in the **Parent Key Column** does not exist in the **Member Key Column**.</li></ul>This property is available for parent-child dimensions only.
	Skipped Levels Column	Indicates the column that contains the number of levels between a member and its parent, excluding the member and parent. Valid values in the column are 0 (zero) and positive integers. This property is available only for parent-child dimensions.
	Unary Operators	Enables unary operators (custom rollup operators) that control how level members are aggregated into their parent member's value. For more information, see Custom Rollup Operators. To view the custom rollup operators, on the **View** menu click **Data**, and then see the value beside

		UNARY_COLUMN in the member properties pane.
	Visible	Indicates whether the level is visible to end users as they browse cubes. To set this property to **False**, you must first set the **Member Keys Unique** property of all lower levels to **True**. The **Visible** property is not available for parent-child dimensions. This property's value can be overridden for individual cubes by the **Visible** property in Cube Editor.
Member property	**Caption**	Contains the caption used to display the member property.
	Data Size	Indicates the maximum number of characters allowed in the column that stores the member property values.
	Data Type	Indicates the data type of the column that stores the member property values.
	Description	Contains a description of the member property.
	Language	Indicates the language used to display the member property.
	Name	Indicates the name of the member property.
	Source Column	Indicates the column that stores the values of the member property. This column must be in one of the dimension tables for the dimension that contains the member property.
	Type	Provides an indicator to client applications of the type of information in the member property values.
	Visible	Indicates whether the member property is visible to end users as they browse the cube.

Schema Tab (Dimension Editor Schema View)

Use the **Schema** tab to display the dimension tables. Joins are indicated by lines connecting the key columns between the tables.

In the **Schema** tab, you can organize the structure of the dimension in a graphical display, add additional tables and levels, and browse the data in the dimension tables.

To view all columns in a table, lengthen the table window vertically, or use the scroll bars. You can also widen the table window to view long column names.

Note Changing the structure of a dimension causes the results of previous dimension processing to be invalidated. After the structure of a dimension has changed, you must process the dimension to re-create the data.

Working in the Schema Tab

This section lists the actions that can be performed in the **Schema** tab. Many actions can be performed in multiple ways. For example, an action can be accomplished by a drag-and-drop procedure, or by right-clicking an object, and then clicking an item on the shortcut menu.

To add a new table

- Right-click anywhere in the **Schema** tab, and then click **Insert Tables**.

To remove a table

- Right-click the title bar of a table, and then click **Remove**.

To replace a table

- Right-click the title bar of the table, and then click **Replace**.

To browse the data in a table

- Right-click the title bar of a table, and then click **Browse Data**.

To create a join between two tables

- Select the key column name in the first table, and then drag it over the corresponding key column name in the second table.

To remove a join between two tables

- Right-click the line depicting the join, and then click **Remove**.

To change the alias of a table

- Right-click the title bar of the table, and then click **Change Alias**.

Dimension Manager

Use this tool to add or remove dimensions from a cube or create new dimensions.

Dimension Manager appears when in Cube Editor, you right-click the Dimensions folder or a dimension and then click **Existing Dimensions**.

You can choose from previously defined dimensions or start the Dimension Wizard to create a new dimension. If you create a new dimension, you can define it as either a shared dimension or private dimension (available only to the cube you are creating).

The previously defined, available dimensions are listed under **Shared dimensions**.

For more information about dimensions, see Dimensions.

Options

Shared dimensions

Select dimensions to add to the cube.

Cube dimensions

View the list of dimensions you have selected.

New Dimension

Click to start the Dimension Wizard, which helps you create a new dimension.

Drillthrough Options Dialog Box

Use this dialog box to specify drillthrough options for a cube or partition.

Drillthrough is an action in which an end user selects a single cube cell and retrieves a result set from the source data for that cell. If drillthrough is enabled, administrators can also drill through in Cube Browser and the Cube Editor **Data** tab. For more information about drillthrough, see Specifying Drillthrough Options.

In this dialog box, if you are editing a cube, you can enable or disable drillthrough for the cube. If you are editing a cube or partition, you can select the columns that are displayed in the result set. You can also specify a filter to limit the rows in the result set.

This dialog box appears when you perform one of the following actions:

- In Cube Editor, on the **Tools** menu, click **Drillthrough Options**. In this case, the drillthrough options apply to a cube.

- In the **Advanced Settings** dialog box, click **Drillthrough**. (The **Advanced Settings** dialog box is displayed when you click **Advanced** in the **Convert to Partition** dialog box or the last step of the Partition Wizard.) In this case, the drillthrough options apply to a partition.

- The **Drillthrough Options** dialog box has two tabs. For more details about the tabs, click a link in the following table.

To do this	See
Select the columns that are displayed when a drillthrough is executed.	Columns Tab (Drillthrough Options Dialog Box)
Specify a filter (WHERE clause expression of an SQL SELECT statement) to limit the result set	Filter Tab (Drillthrough Options Dialog Box)

returned by drillthrough.	

Options

The following options are displayed for all tabs.

Enable drillthrough

Select to enable end users to drill through any cube cell they are authorized to access. This action also allows administrators to drill through. In the **Columns** tab, you must select at least one column.

The only end users who can drill through are those in cube roles granted the ability to drill through. For more information, see Creating Cube Roles.

Clear to disable end users' and administrators' ability to drill through.

Enable drillthrough is read-only if you are editing a partition. Drillthrough can be enabled and disabled only at the cube level.

OK

Click to temporarily save your changes and close the dialog box. To permanently save your changes:

- If you are editing a cube, in Cube Editor, save the cube.

- If you are editing a partition, in the **Advanced Settings** dialog box, click **OK**. Then, in the Partition Wizard, click **Finish** (or, in the **Convert to Partition** dialog box, click **OK**).

Columns Tab (Drillthrough Options Dialog Box)

Use this tab to select the columns that are displayed when a drillthrough is executed.

IMPORTANT If you select a column to which a cube role is denied access and then grant drillthrough permissions to the cube role, you will create a security exposure. (You can deny access to a column by using either dimension security or cell security. For more information, see Dimension Security and Cell Security.) When the end users in the cube role drill through, they will access the denied column.

If you are editing a cube, options in this tab can be overridden in the cube's partitions. For information about accessing the drillthrough options of a partition, see Specifying Drillthrough Options.

Options

Column

Select the columns that are displayed in the result set when a drillthrough is executed. By default only the tables in the cube's or partition's schema are displayed. To add other tables in the cube's or partition's data source, click **Add Table**, and then use the **Select Table** dialog box.

Table

View the tables in which the columns are defined.

In multiple-partition cubes, fact table names can vary among the partitions. However, if you are editing a multiple-partition cube, the **Columns** tab displays only the columns and tables in the cube's default partition. For drillthrough in a multiple-partition cube, Microsoft® SQL Server™ 2000 Analysis Services automatically changes the fact table name, if necessary, for each partition.

Add Table

Click to display the **Select Table** dialog box, where you can add a table and its columns to the list. You can select tables only from the data source of the cube or of the partition.

Select All

Click to select all columns.

Deselect All

Click to clear the selections of all columns.

Filter Tab (Drillthrough Options Dialog Box)

Use this tab to specify a filter (WHERE clause expression of an SQL SELECT statement) to limit the result set returned by drillthrough. Type the text that follows WHERE, but exclude the WHERE keyword.

For example, to limit the result set to 1998 data, type:

```
"the_year"=1998
```

Another example (alphanumeric data):

```
"the_month"='January'
```

Whenever necessary to avoid ambiguity, use a qualified expression. For example, if a column name appears in multiple tables, include the table name in the expression:

```
"time"."the_month"='January'
```

The SELECT statement used to retrieve rows for the creation of the result set is generated automatically. The filter expression is connected with an AND to the automatically generated part of the WHERE clause. Therefore, if you specify multiple filter expressions, enclose them all in a pair of parentheses, for example:

```
("the_year"=1998 OR "the_year"=1999)
```

The filter expression acts as a pass-through statement, and its syntax is not checked until a drillthrough is executed. If the syntax is incorrect, drillthrough fails.

For more information, see Drillthrough Filters.

If you are editing a cube, the filter in this tab can be overridden in the cube's partitions. For information about accessing the drillthrough filter of a partition, see Specifying Drillthrough Options.

Edit Virtual Dimension Dialog Box

Use this dialog box to select, from an existing dimension, the new member property on which the virtual dimension will be based.

This dialog box is displayed when, in the Analysis Manager tree pane, you right-click a virtual dimension created in Microsoft® SQL Server™ version 7.0 OLAP Services, and then click **Edit**. If you perform the same action on a virtual dimension created in SQL Server 2000 Analysis Services, Dimension Editor is displayed. For more information about virtual dimensions created in SQL Server 7.0 OLAP Services, see Virtual Dimensions Created in Version 7.0.

Options

Member properties

Select the member property on which you want to base your virtual dimension.

Filter Expression Dialog Box

Use this dialog box to specify a filter to limit the fact table records used in the incremental update of the cube or partition you selected in the **Analysis Services Processing Task** dialog box.

This dialog box is displayed when, in the **Analysis Services Processing Task** dialog box, you select a cube, click **Incremental update**, and then click the edit (**...**) button beside the **Filter** box.

Caution If in the **Analysis Services Processing Task** dialog box you select the default fact table (that is, the same fact table already used by the cube or partition), you must create a filter to ensure that only data not already in the cube or partition will be added. Otherwise, the cube or partition will contain duplicate and therefore inaccurate data.

For example, a cube contains data for years 1995 through 1997. You are adding data for 1998 from the same table that supplies the 1995 through 1997 data. The name of the column that contains years is **the_year**. You must use the following filter expression:

```
"the_year"=1998
```

Another example (alphanumeric data):

```
"the_month"='January'
```

Options

Filter expression

Type a WHERE clause expression of an SQL SELECT statement. Do not include the WHERE keyword.

Whenever necessary to avoid ambiguity, use a qualified expression. For example, if a column name appears in multiple tables, include the table name in the expression:

```
"time"."the_month"='January'
```

The SELECT statement used to retrieve records for the incremental update is generated automatically. The filter expression is connected with AND to the automatically generated part of the WHERE clause. Therefore, if you specify multiple filter expressions, enclose them all in a pair of parentheses. For example, to retrieve records for years 1998 and 1999, use the following filter:

```
("the_year"=1998 OR "the_year"=1999)
```

Filters consist of one or more expressions using columns in the fact table. A filter can also contain columns in dimension tables if they are included in a nested SELECT statement and the underlying database supports nested SELECT statements.

The filter expression acts as a pass-through statement, and its syntax is not checked until the package is executed. If the syntax is incorrect, the incremental update and the task fail.

See Also

Analysis Services Processing Task Dialog Box

Partition Filters and Incremental Update Filters

Find Node Dialog Box

Use this dialog box in Dependency Network Browser to select a node by name. In large data mining models, some nodes may be excluded in the initial presentation. Other nodes may be difficult to locate. Using the **Find Node** dialog box provides a way to quickly locate or add nodes to the network browser pane. You can only select or add one node at a time.

This dialog box is displayed when, on the Dependency Network Browser toolbar, you click the **Find** tool. For more information, see Toolbar (Dependency Network Browser).

Options

Node Name

Type the name of the node you want to select. As you type, the list of available nodes narrows to match the values you provide.

Show hidden nodes

Select this option to display nodes that are not visible in the network browser pane. You can select this option when you want to add nodes that are not included in the initial presentation of the data mining model.

Node List

Click the name of the node that you want to add to the network browser pane. You can also double-click a name to simultaneously select and close the dialog box. This list displays all nodes defined in the data mining model.

See Also

Dependency Network Browser

Viewing with Dependency Network Browser

Import Actions Dialog Box

Use this dialog box to import an existing action into a virtual cube from another cube that is within the same database. After selecting an individual action to import, you can view the Multidimensional Expressions (MDX) statement that was used to create the action in the **Action MDX** box.

This dialog box appears for a virtual cube when, in Virtual Cube Editor, you perform one of the following actions:

- On the **Edit** menu, select **Import**, and then click **Actions**.
- In the tree pane, right-click the **Actions** folder, and then click **Import Action**.
- On the toolbar, click **Import Actions**.

Options

Available Actions

To select an individual action from a cube, select the check box next to the name of the action you want to select. To select all actions from a cube, select the check box next to the name of the cube.

Show source cubes only

Clear this check box to display all available cubes and source cubes in the **Available Actions** tree view.

Import Calculated Cells Dialog Box

Use this dialog box to import a calculated cells definition into a virtual cube from its component cubes.

This dialog box is displayed in Virtual Cube Editor when you perform one of the following actions:

- On the **Edit** menu, click **Import**, and then click **Calculated Cells**.
- Click the **Import Calculated Cells** button on the toolbar.
- In the tree pane of the Virtual Cube Editor, right-click the Calculated Cells folder, and then click **Import Calculated Cells**.

Options

Available Calculated Cells

Select the check boxes beside the calculated cells definitions that you want to import. If you want to import all the calculated cells definitions in a cube, select the check box next to the cube.

This box lists only calculated cells definitions that are in the component cubes of the virtual cube.

If you select a calculated cells definition with the same name as a calculated cells definition already in the virtual cube, the selected calculated cells definition is imported with a new name consisting of its old name and a numeric suffix.

Value expression

View the calculation formula for the selected calculated cells definition. To change this formula in the virtual cube, import the calculated cells definition and edit it in the Calculated Cells Wizard.

Show source cubes only

Specifies whether to view calculated cells definitions only in the component cubes of the virtual cube. Clear this check box to view all calculated cells definitions in the database. Adding a calculated cells definition that is not in a source cube of the virtual cube may generate formula errors. After you import a calculated cells definition, you can edit it to fix formula errors or make any other required changes.

See Also

Calculated Cells

How to import calculated cells into a virtual cube

How to edit a calculated cells definition

Import Calculated Members Dialog Box

Use this dialog box to import calculated members into a virtual cube from its component cubes.

This dialog box is displayed when, in Virtual Cube Editor, you perform one of the following actions:

- On the **Edit** menu, click **Import**, and then click **Calculated Members**.
- Click the **Import Calculated Members** button on the toolbar.
- In the tree pane, right-click the Calculated Members folder, and then click **Import Calculated Member**.

Options

Available Calculated Members

Select the check boxes beside the calculated members you want to import. If you want to import all the calculated members in a cube, select the check box next to the cube.

If you select a calculated member with the same name as a calculated member already in the virtual cube, the selected calculated member is imported with a new name consisting of its old name and a numeric suffix.

Value expression

View the expression for the selected calculated member. To change this expression in the virtual cube, import the calculated member and edit it in Calculated Member Builder.

Show source cubes only

Specifies whether to view calculated member definitions only in the component cubes of the virtual cube. Clear this check box to select from all calculated members in the database. Adding a calculated member that is not in a source cube of the virtual cube may generate formula errors. After you import calculated members, you can edit them to fix formula errors or make any other necessary changes.

Import Named Sets Dialog Box

Use this dialog box to import an existing named set into a virtual cube from another cube that is within the same database. After selecting an individual named set to import, you can view the Multidimensional Expressions (MDX) statement that was used to create the named set in the **Set Expression** box.

This dialog box appears for a virtual cube when, in Virtual Cube Editor, you perform one of the following actions:

- On the **Edit** menu, point to **Import**, and then click **Named Sets**.
- In the tree pane, right-click the **Named Sets** folder, and then click **Import Named Set**.
- On the toolbar, click **Import Named Sets**.

Options

Available Named Sets

To select an individual named set from a cube, select the check box next to the name of the desired named set. To select all named sets from a cube, select the check box next to the name of the cube.

Show source cubes only

Clear this check box to display all available cubes in the **Available Named Sets** tree view as well as the virtual cube's source cubes.

Insert Level Dialog Box

Use this dialog box to select a column for a dimension in Dimension Editor or Cube Editor.

This dialog box appears at various times when you edit dimensions in Cube Editor or Dimension Editor.

Options

Tree box

View the available columns.

Column Types

Select or clear check boxes to display columns of the type described.

Insert Measure Dialog Box

Use this dialog box to select a source column for a measure in Cube Editor.

This dialog box appears for a cube when, in Cube Editor, you perform one of the following actions:

- On the **Insert** menu, click **Measure**.
- In the tree pane, right-click the cube icon, and then click **New Measure**.
- On the toolbar, click the **Insert Measure** button.

Options

Tree box

View the available columns.

Column Types

Select or clear check boxes to display columns of the type described.

Insert Member Property Dialog Box

Use this dialog box to select a column for a new member property of a level in Dimension Editor or Cube Editor.

This dialog box appears when you create a new member property in Cube Editor or Dimension Editor.

Options

Tree box

View the available columns.

Column Types

Select or clear check boxes to display columns of the type described.

Join Columns Dialog Box

Use this box to create joins between tables in a cube's schema.

This dialog box appears when, in the schema view of Cube Editor or Relational Mining Model Editor, you right-click a column in a table, and then click **Insert Join**.

Options

Select a column

Choose a column from a table to join to the originally selected column.

Column Types

Select a data type to restrict the selection of available columns.

OK

Click to create the join and to close the dialog box.

Level Naming Template Dialog Box

Use this dialog box to specify the level names that are displayed to end users when they browse a cube containing the dimension displayed in the dialog box title bar.

This dialog box is displayed only for parent-child dimensions. A parent-child dimension always contains a single meta data level (excluding the (All) level, if any), which typically produces multiple displayed levels. The number of displayed levels is not always known when the dimension is created or edited. Also, this number can change when the data in the dimension table is updated. For these reasons, you can use this dialog box to specify the names applied to the displayed levels.

For more information about parent-child dimensions, see Parent-Child Dimensions.

If you do not specify level names, by default the displayed levels are named Level *nn* where *nn* is an integer indicating the level's rank, starting from the top level. An exception is the (All) level, which must be named (All).

This dialog box appears when, in the tree pane of Dimension Editor or Cube Editor, you select the level in a parent-child dimension, and, in the **Advanced** tab of the properties pane, click **Level Naming Template**, and then click the edit (**...**) button.

After you change values in this dialog box for a shared dimension and save it, you must process it with the **Incremental update** option. For more information, see Updating and Rebuilding Shared Dimensions. After you change values in this dialog box for a private dimension and save the cube that contains it, you must process the cube with the **Refresh data** option. For more information, see Updating and Refreshing Cube Data.

Options

Level

View the displayed level names ranked from highest to lowest. By default, if the dimension's **All Level** property is **No**, only the row for Level 1 is displayed. If the dimension's **All Level** property is **Yes**, only the rows for Level 1 and Level 2 are displayed.

You can add a blank row by placing your cursor in the last row's **Name** column and typing a value. Each row's value in the **Level** column is one greater than the preceding row's. You cannot edit the **Level** values.

If the dimension's **All Level** property is **Yes**, the entire row for the (All) level (**Level** 1) is read-only because the name of the (All) level must be (All). (Some client applications do not display the (All) level's name.)

Name

View the level names displayed to end users. There are two common techniques for supplying the level names:

- In the top row (or second row if the dimension's **All Level** property is **Yes**), type a value followed by a space and asterisk (*). This technique produces names that are identical except for an incrementing, numeric suffix, which replaces the asterisk. For example, if the dimension's **All Level** property is **No**, in the top row type My Level * to apply the name My Level 01 to the top level, the name My Level 02 to the next level, and so on. Or, if the dimension's **All Level** property is **Yes**, in the second row type My Level * to apply the name My Level 02 to the level below the (All) level, the name My Level 03 to the next level, and so on.

- Add rows and type a level name in each. If the number of supplied level names is exceeded by the number of displayed levels, the last nonempty **Name** value along with an appended, incrementing, numeric suffix is applied to the unnamed levels. For example, if the last **Name** value, supplied for Level 8, is Element, the names Element 01, Element 02, and so on are applied to the unnamed levels. If the last **Name** value contains an asterisk, the resulting suffixes indicate each level's rank, starting from the top level. If the last **Name** value, supplied for Level 8, is Element *, the name Element 08 is applied to **Level** 8, and the names Element 09, Element 10, and so on are applied to the unnamed levels.

The **Result** value changes to reflect the current values in the **Name** column. To refresh the **Result** value, click anywhere in the **Level** or **Name** column except in the row where you most recently typed.

To display an asterisk in the level name, type a double asterisk (**) in the **Name** column.

Because semicolons (;) are used to separate level names when they are stored, it is recommended that you omit them from level names. If you type a semicolon in the **Name** column, it is interpreted as a level name separator.

Clear All

Click to delete all values in the **Level** and **Name** columns except the values for the (All) level, if any.

Result

View the first five level names that are displayed to end users. The names are separated by semicolons (;). The names change to reflect the values in the **Name** column. The semicolons and ellipses (...) are not displayed to end users.

OK

Click to temporarily save your changes and close the dialog box. To permanently save your changes, in the editor, on the **File** menu, click **Save**.

Linked Cube Dialog Box

Use this dialog box to create a linked cube. A linked cube is based on another cube, called a *source cube*, which is defined and stored on another Analysis server. You can create linked cubes as an alternative to storing multiple copies of a cube on multiple Analysis servers. For more information about linked cubes, see Linked Cubes.

To create a linked cube, you must have access to the source cube. You have access if your user name is included in either of the following on the Analysis server where the source cube is defined: the OLAP Administrators group or a cube role for the source cube.

This dialog box appears when, in the Analysis Manager tree pane, you right-click a Cubes folder and then click **New Linked Cube**.

Options

Name

Type a name for the linked cube. The name must begin with an alphabetical character and can be up to 50 characters in length.

Source Cube

Expand a data source and select the source cube.

IMPORTANT Do not select from a data source that has the same Analysis server as the Analysis server on which you are creating the linked cube.

Cubes are displayed only if they have been processed since they were last changed.

You can refresh the connection to a data source by right-clicking the data source and then clicking **Refresh Connection**.

Dimensions

View the dimensions contained in the cube selected in the **Source Cube** box. The **Dimensions** box is read-only.

New Data Source

Click to display the **Multidimensional Data Source** dialog box, where you can specify a new multidimensional data source. For more information about this procedure, see Managing Linked Cubes.

Process after creating new cube

Select to begin processing the new linked cube when you click **OK**.

To process a linked cube on an Analysis server, the Analysis server service (MSSQLServerOLAPService) logon account must be a domain user account. This account must have access to the source cube. The account has access if it is included in either of the following on the publishing server: the OLAP Administrators group or a cube role for the source cube. If the account does not have access to the source cube, computers or users cannot process the linked cube through that account.

Map the Column Dialog Box

Use this dialog box to create a new level in a dimension, new dimension, or new measure in Cube Editor.

This dialog box appears when, in the Cube Editor schema view, you double-click a column in a table.

Options

Level in Dimension

Select to create a new level in the private dimension you select from the drop-down list. This option is available only when the cube contains a private dimension.

Dimension

Select to create a new private dimension.

Measure

Select to create a new measure from the fact table of this cube.

MDX Builder

Use this dialog box to create an expression in the Multidimensional Expressions (MDX) syntax. For information about the MDX functions in Microsoft® SQL Server™ 2000 Analysis Services, see MDX Function List.

This dialog box is displayed from dialog boxes and wizards in which an MDX expression can be specified.

Options

MDX expression

Build the MDX expression using the **Data** box, **Functions** box, **Insert** button, arithmetic operator buttons, and number buttons.

Note The **Data** and **Functions** boxes provide a convenient means for building valid expressions. You can double-click or drag values from both boxes to the **MDX expression** box.

You can also type an expression directly into the **MDX expression** box. You may need to use this method if you want to add functions from libraries other than the Analysis Services MDX function library.

Data

View the data structures in the object to which the MDX expression applies.

Functions

View the functions in the Analysis Services MDX function library and any additional libraries that have been registered. For information about an individual Analysis Services MDX function, select it, and then press F1.

To add a function to the **MDX expression** box, place the cursor where you want to insert the function, select a function in the **Functions** box, and either double-click or click **Insert**. The function syntax appears in the **MDX expression** box. Replace arguments and their delimiters (« and ») with the appropriate values.

Insert

Click to add the item selected in the **Data** box or **Functions** box to the MDX expression.

Arithmetic operator buttons

Click to add operators to the MDX expression.

Number buttons

Click to add numbers to the MDX expression.

Register

Click to display the **Register Function Libraries** dialog box, where you can register an external function type library (*.olb, *.tlb, *.dll). The new library appears in the **Functions** box. For more information, see Register Function Libraries Dialog Box.

Usage Notes

You can use the following techniques when you create MDX expressions:

- **Null** keyword

  You can use the keyword **Null** to create an MDX expression that has a null value.

- Temporary alias for a set

  A set in a function can be given a temporary alias for use within the function by using the keyword **As**. In this example, the set {State} is given the temporary alias S1 so it can be referred to later in the function in a nested iteration:

  ```
  Sum({State} As S1, Sum(Geography.CurrentMember.Children
  Population * Val((Geography.CurrentMember.Properties("City
  Val(S1.Current.Properties("StateTax")))))
  ```

Mining Model Role Dialog Box

Use this dialog box to create or edit a mining model role.

This dialog box is displayed when you perform any of the following actions:

- In Mining Model Role Manager, click **New**.
- In Mining Role Manager, right-click an entry in the **Role** column, the **Membership** column, or the **Description** column, and then click **New**.
- In the Analysis Manager tree pane, right-click a mining model roles icon, click **Manage Roles**, and then click **New**. The mining model roles icon appears here.

Note The default values for a mining model role are derived from the database role of the same name. Changes you make in this role are also made in the database role of the same name.

Options

Role name

View or edit the name of the role. You can enter a maximum of 50 characters; the name must begin with an alphabetical character. If you are editing an existing role, this box displays the name of that role, and you cannot change it.

Description

View or edit the description of the role. You can enter a maximum of 200 characters. If you are editing an existing role, you cannot change the description.

Membership tab

View the names of users and groups of the mining model role and their associated domains.

Add

Click to display the **Add Users and Groups** dialog box, where you can add Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 users and groups to the membership of the role.

Remove

Click to delete the selected role.

Mining Model Role Manager

Use this tool to create, edit, and maintain mining model roles.

A mining model role applies to only a single model. A role is created when you assign a role to a mining model by selecting the role in Mining Model Role Manager. This grants the role's users access to the model. The name and default values of a model role are derived from the database role with the same name.

In Mining Model Role Manager, each row whose check box is selected in the list displays a role. Users in these roles can access the model whose name appears in the title bar. These roles can be maintained within Mining Model Role Manager.

Each unchecked row displays a role not assigned to the mining model. Only users who are members of the selected roles can access the model.

To grant users in a database role access to the model, select the check box beside the role. To assign a role to the mining model, select the check box beside the role. This creates a mining model role with the same name as the database role.

The roles assigned to the model are listed first, followed by the roles that are not assigned to mining models (that is, the remaining database roles). Each category is sorted alphabetically by the role names in the **Role** column.

Mining Model Role Manager appears when, in the Analysis Manager tree pane, you right-click either a mining model roles icon or a mining model icon, and then click **Manage Roles**.

Options

Access

Select a check box next to a role to determine whether it can access the model listed in the title bar of Mining Model Role Manager. Only roles whose check boxes are selected can access the model.

Role

View role names.

Membership

Modify the Microsoft® Windows NT® 4.0 or Windows® 2000 users and groups in each role.

To change the membership of a mining model role, right-click the **Membership** column and click **Edit**, or click the edit (**...**) button to display the **Mining Model Role** dialog box.

Note Changes in this tab propagate to the database role and cube roles with the same name as the edited mining model role.

Description

View the description of each role.

Show

Use to limit the roles displayed in the list. By default, **All roles** is displayed, but you can limit by a user name or group name in the roles. To limit by a user name or group name in the roles, in the box beside **Show**, select **Roles containing users**. In the next box, type the user name or group name, or type part of a name, and then click the magnifying glass button.

New

Click to display the **Mining Model Role** dialog box, where you can create and add a new mining model role. When you create a new role, a database role with the same name and specifications is also created.

Edit

Click to display the **Mining Model Role** dialog box, where you can edit this role.

Duplicate

Click to display the **Duplicate Role** dialog box, where you can supply a name for a new role based on the selected mining model role. When you create a new mining model role in this way, a database role with the same name and specifications is also created.

Merge Partitions Dialog Box

Use this dialog box to merge two partitions. The data and aggregations of the partition you selected in the Analysis Manager tree pane are consolidated into the partition you select in the **Into this partition** box, and the first partition is then deleted.

This dialog box appears when, in the Analysis Manager tree pane, you expand the Partitions folder for a cube, right-click a partition, and then click **Merge**. This command is available only for partitions in cubes that contain more than one partition.

Caution When you merge partitions that have different fact tables, you must manually merge the fact tables so the target partition's fact table contains all the facts required for the combined partition. For more information, see Merging Partitions.

Options

Merge this partition

View the partition you selected in the Analysis Manager tree pane. This is the source partition. After being merged, it is deleted.

into this partition

Select the target partition with which you want the source partition to be merged. After merging, this partition is the one that remains.

Merge

Click to consolidate the specified partitions. The newly merged partition is processed after you click **Merge**. Processing status of the partition is displayed in the **Process** dialog box.

Multidimensional Data Source Dialog Box

Use this dialog box to define a new multidimensional data source for a new linked cube. This new data source must use an existing Microsoft® SQL Server 2000™ Analysis Services database on an Analysis server, and you must have access to the database. You can provide access by including your user name in the OLAP Administrators group for the linked cube's publishing server or by including it in a cube role for the source cube.

For more information about data sources, see Data Sources. For more information about access requirements for linked cubes, see Linked Cubes.

This dialog box appears when in the **Linked Cube** dialog box, you click **New Data Source**.

Options

Server

Type the name of the Analysis server to which you want to connect.

Note You cannot connect to an Analysis server by specifying its IP address.

Database

Select the database that contains the source cube you want to use for the new linked cube.

Advanced

Click to display the **Data Link Properties** dialog box. Use this dialog box to specify special options to connect to the linked cube's publishing server.

Test Connection

Click to verify your connection to the linked cube's publishing server and source cube.

Named Set Builder

Use this tool to create an expression to define a named set in the Multidimensional Expressions (MDX) syntax. You can build the MDX expression for the named set by using appropriate combinations of the **Data** box, **Functions** box, **Insert** button, arithmetic operator buttons, and number buttons, or by typing an expression directly into the **Set expression** box.

This tool appears when you are creating a new named set in either Cube Editor or Virtual Cube Editor.

Options

Set name

Enter a name for the new named set.

Set expression

Type the MDX expression to define the named set.

IMPORTANT The **Data** and **Functions** boxes provide a convenient means for building valid expressions. You can either double-click or drag values from both boxes to add them to the **Set expression** box. However, to add functions from libraries other than the Microsoft® SQL Server™ 2000 Analysis Services MDX function library, you may need to type expressions containing such functions directly into the **Set expression** box.

Data

Select the data structures in the cube that can be used as elements of the MDX expression.

Functions

Select the functions in the Analysis Services MDX function library and any additional libraries that have been registered. For more information about the functions, see Function List. For information about an individual Analysis Services MDX function, select it, and then press F1.

To add a function to the **Set expression** box, place the cursor where you want to insert the function, select a function in the **Functions** box, and click **Insert**. The function syntax appears in the **Set expression** box. Replace arguments and their delimiters (« and ») with the appropriate values.

Insert

Click to add the item selected in the **Data** box or **Functions** box to the **Set expression** box.

Arithmetic operator buttons

Click to add operators to the **Set expression** box.

Number buttons

Click to add numbers to the **Set expression** box.

Register

Click to display the **Register Function Libraries** dialog box, where you can register an external function type library (*.olb, *.tlb, *.dll). The new library will appear in the **Functions** box. For more information, see Register Function Libraries Dialog Box.

MDX Usage Notes

You can use the following techniques when you create MDX expressions:

- **Null** keyword

  You can use the keyword **Null** to create an MDX expression whose value is null.

- Temporary alias for a set

  A set in a function can be given a temporary alias for use within the function by using the keyword **As**. In this example, the set {State} is given the temporary alias S1 so it can be referred to later in the function in a nested iteration.

  ```
  Sum({State} As S1, Sum(Geography.CurrentMember.Children
  Population * Val((Geography.CurrentMember.Properties("City
  ```

```
    Val(S1.Current.Properties("StateTax")))))
```

See Also

Named_Sets

OLAP Mining Model Editor

Use this tool to browse the content of an OLAP data mining model and edit its structure. You can also use OLAP Mining Model Editor and the associated dialog boxes to perform various procedures.

The following diagram shows the elements that make up OLAP Mining Model Editor.

Unlike Relational Mining Model Editor, OLAP Mining Model Editor does not show a schema view, because the OLAP data mining model is based on the structure of a single cube. The browser pane shows only the content of the OLAP data mining model, while the structure pane contains the complete structure of the cube, including indicators to show which dimensions, levels, member properties, and measures from the cube provide input and predictable information to the OLAP data mining model.

OLAP Mining Model Editor appears when you do one of the following:

- In the Analysis Manager tree pane, right-click an OLAP data mining model, and then click **Edit**.

- In the Analysis Manager tree pane, click an OLAP data mining model, and then on the **Action** menu, click **Edit**.

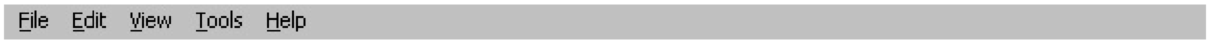
OLAP Mining Model Editor has five elements. For more information about these elements, click a link in the following table.

To do this	See
Perform various basic tasks within OLAP Mining Model Editor.	Menu Bar (OLAP Mining Model Editor)
Carry out common functions within OLAP Mining Model Editor.	Toolbar (OLAP Mining Model Editor)
View the OLAP data mining model structure, such as input and predictable columns.	Structure Pane (OLAP Mining Model Editor)

Change the structure of the OLAP data mining model.	
View and edit properties associated with the OLAP data mining model and its data mining columns. View descriptions of properties and the objects to which they apply.	Properties Pane (OLAP Mining Model Editor)
Display the content of the processed OLAP data mining model.	Browser Pane (OLAP Mining Model Editor)

Menu Bar (OLAP Mining Model Editor)

Use the menu bar in OLAP Mining Model Editor to perform various tasks with OLAP data mining models.

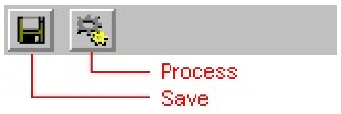


The following table describes the menus and their associated options.

Menu item	Description
File	Saves OLAP data mining models and exits OLAP Mining Model Editor.
Edit	Allows you to include in or remove from the OLAP data mining model dimensions, levels, member properties, and measures associated with the cube upon which the mining model is based, as well as specifying the case dimension for the OLAP data mining model.
View	Provides property viewing options.
Tools	Supports processing of the OLAP data mining model.
Help	Provides access to Help about OLAP Mining Model Editor, as well as to SQL Server Books Online, which contains all of the Analysis Services documentation.

Toolbar (OLAP Mining Model Editor)

Use the toolbar to access the most commonly used functions in OLAP Mining Model Editor.

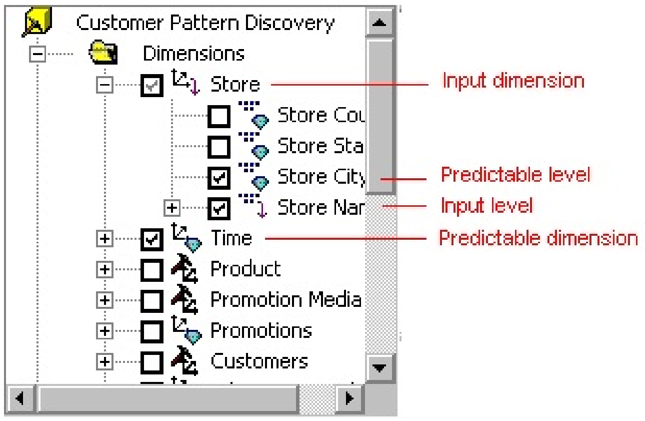


The following table describes the toolbar buttons in greater detail.

Button	Description
Save	Saves the OLAP data mining model.
Process Mining Model	Processes the OLAP data mining model.

Structure Pane (OLAP Mining Model Editor)

Use the structure pane in OLAP Mining Model Editor to review and change the structure of an OLAP data mining model. The structure pane provides a graphical representation of the usage of each dimension, level, member property, and measure in the cube associated with the OLAP data mining model, as shown in the following diagram.



The check boxes beside each cube element in the structure pane indicate whether the cube element is used to define the OLAP data mining model. Depending on the setting of the **Case Dimension**, **Case Level**, or **Usage** properties, the checked cube elements are used to create data mining columns of various usage types.

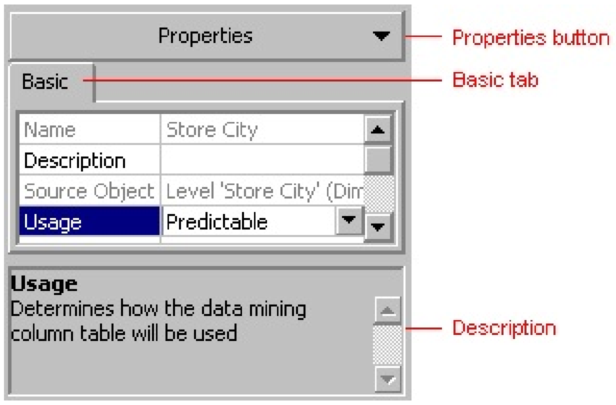
The following table describes the data mining column usage types in more detail.

Column type	Description
Input	The dimension, level, member property, or measure is used by the OLAP data mining model as an attribute column, to supply input for the case set.
Predictable	The dimension, level, member property, or measure is used by the OLAP data mining model as a

	predictive column, to supply output for the case set.
Input and predictable	The dimension, level, member property, or measure is used by the OLAP data mining model as an attribute column, to supply both input and output for the case set.

Properties Pane (OLAP Mining Model Editor)

Use the properties pane in OLAP Mining Model Editor to view properties associated with the OLAP data mining model and its data mining columns.



Name	Store City
Description	
Source Object	Level 'Store City' (Dim
Usage	Predictable

Usage
Determines how the data mining column table will be used

The following table describes the features of the properties pane.

Feature	Description
Properties button	Use this button to show or hide the properties pane.
Basic tab	The **Basic** tab shows the most commonly used properties, such as **Name** and **Description**, for the OLAP data mining model and mining model columns.
Description	This area of the properties pane displays the name and brief explanation of a property selected in the **Basic** tab.

Basic Properties

The **Basic** tab is used to display and, optionally, edit the most commonly viewed properties for OLAP data mining models and data mining columns. The following table describes the properties displayed in the **Basic** tab in more detail for each type of object, such as mining model, dimension, level, member property, and measure, that can be selected in the structure pane.

		Applicable

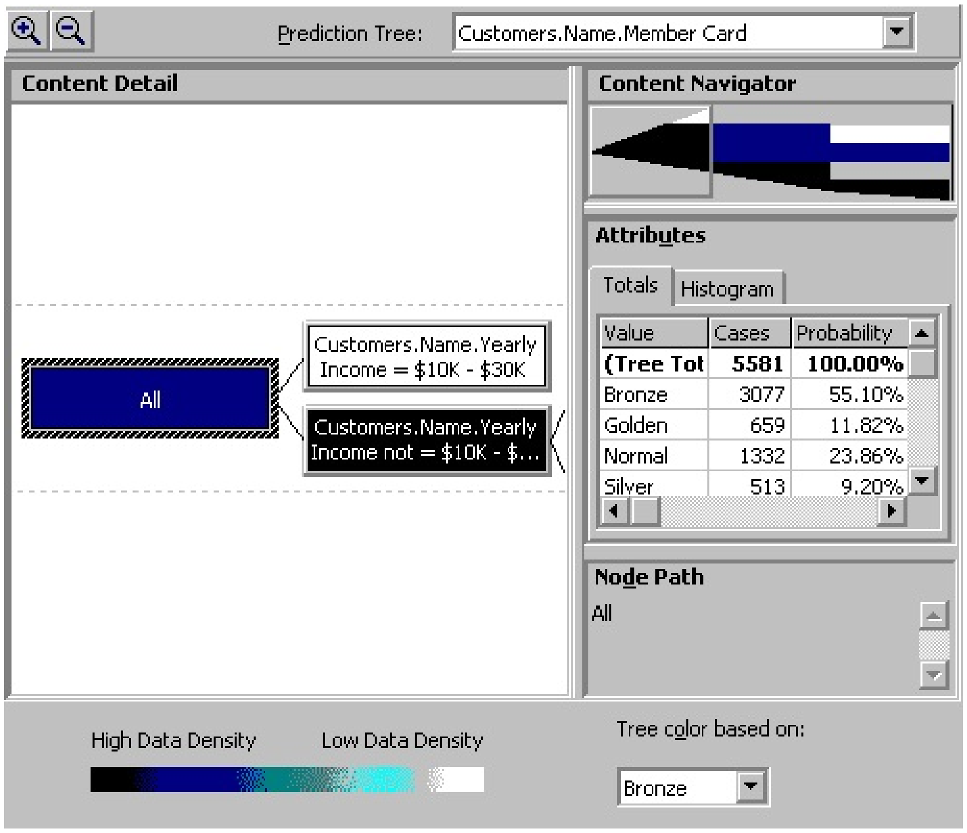
Property	Description	objects
Name	The name of the selected object. This property is read-only.	All
Description	The description of the selected object. This property is not available for dimensions whose **Case Dimension** property is **True**, and is read-only for levels whose **Case Level** property is **True**.	All
Source Object	The source object from the cube associated with the OLAP data mining model.	All, except data mining model
Usage	Determines how the object will be used in the OLAP data mining model. For more information about the usage types, see structure pane (OLAP Mining Model Editor). This property is not available for dimensions whose **Case Dimension** property is **True**, and is read-only for levels whose **Case Level** property is **True**.	All
Source Cube	The source cube associated with the OLAP data mining model.	Data mining model
Mining Algorithm	The name of the mining algorithm provider used by the OLAP data mining model.	Data mining model
Additional Parameters	A comma-delimited list of mining parameters and values for the OLAP data mining model.	Data mining model
Case Dimension	Indicates whether or not the dimension provides the case set to be used by the OLAP data mining model.	Dimension
Case Level	Indicates whether or not the level provides the case set to be used by	Level

	the OLAP data mining model.	
Cluster Count	The number of clusters the algorithm identifies.	Data mining model

Browser Pane (OLAP Mining Model Editor)

The browser pane of OLAP Mining Model Editor shows the mining model content of processed OLAP data mining models. The browser pane uses Data Mining Model Browser to illustrate the mining model content.

The following diagram shows the browser pane for processed OLAP data mining models, with a displayed Microsoft® Decision Trees node structure.



Prediction Tree: Customers.Name.Member Card

All

Customers.Name.Yearly Income = $10K - $30K

Customers.Name.Yearly Income not = $10K - $...

Value	Cases	Probability
(Tree Tot	**5581**	**100.00%**
Bronze	3077	55.10%
Golden	659	11.82%
Normal	1332	23.86%
Silver	513	9.20%

Node Path: All

Tree color based on: Bronze

For more information on the Data Mining Model Browser, see Data Mining Model Browser.

Partition Processing Settings Dialog Box

Use this dialog box to specify settings for processing a cube. Settings in this dialog box are used whenever a cube or one of its partitions is processed.

This dialog box appears when, in DTS Designer, you click **Cube Settings** in the **Analysis Services Processing Task** dialog box.

IMPORTANT Cube or partition processing options are stored with the meta data for the cube or partition. Changes to the settings for these options replace previous settings for the cube or partition, regardless of whether the options are set by Analysis Manager, Data Transformation Services (DTS) tasks, or Decision Support Objects (DSO) applications.

Options

After all aggregations have been calculated

Select to make data unavailable for browsing until all aggregations have been computed. This is the default option.

Immediately after data is loaded

Select to make data available for browsing after it has been loaded but before all aggregations have been computed. This means that new data is available for querying sooner, but the Analysis server may not be as responsive to query execution.

Stop processing after encountering missing dimension key errors

Select to stop processing if dimension key errors are encountered.

- **Processing will stop after**

  Specify the number of errors (the default is 0).

Ignore all missing dimension key errors

Select to have processing continue despite dimension key errors.

Log dimension key errors to a file

Select to log dimension key errors to a file. Use one of the following options to provide the file name and path.

- **File path and name**

  Type a path to the text file in which you want to log dimension key errors.

- **Browse**

  Click to select a local or network location for the file.

See Also

Process a Cube Dialog Box

Processing Cubes

Prediction Query Builder

Use this tool to create prediction query syntax for a Data Mining Prediction Query task that conforms to the OLE DB for Data Mining specification.

This dialog box is displayed when you click **New Query** on the **Query** tab of the **Data Mining Prediction Query Task** dialog box.

Options

Case table

Select the source table that contains the cases to be analyzed for the prediction query.

Browse

Click to view a sample of data from the source table.

Input columns

Clear the check box next to the name of an input column other than the case key column to remove it from the prediction query. The case key column is used to uniquely identify records in the results of the prediction query.

Note Excluding columns from **Input columns** may affect the values of the predicted column in the output table the task produces.

Name

View the names of input columns from the source table that are available to be included in the prediction query.

Source Column

View the names of input columns selected to be included in the prediction query.

Predicted column

Select a predicted column from the list of available prediction columns. The definition of the data mining model used in the data mining prediction query task determines which columns are available.

See Also

Data Mining Prediction Query Task Dialog Box

Predictions and Results of Data Mining

Process Dialog Box

Use this dialog box to monitor processing operations while processing your cube, dimension, data mining model, or database.

This dialog box appears at various times when you process, incrementally update, or refresh the data in a cube or dimension. It also appears when you process a database or when you process or incrementally update a data mining model.

Options

Status bar

> View the status of processing operations on the cube, dimension, data mining model, or database.

Stop

> Click to halt and cancel processing operations on the cube, dimension, data mining model, or database.

Reprocess

> Click to restart processing operations on the cube, dimension, data mining model, or database from the beginning.

View Details

> Click to display the **View Trace Line** dialog box. Click a message, and then click **View Details** to see details about the message.

Process a Cube Dialog Box

Use this dialog box to choose how to process a cube.

This dialog box appears when you right-click a cube in the Analysis Manager tree pane, and then click **Process**.

Options

Incremental update

Select to append new data from the data warehouse to one of your cube's partitions and update aggregations. An incremental update does not affect the existing data that has already been processed and usually requires significantly less time than processing with the **Process** option.

An incremental update can be performed while users continue to query the cube; after the update has completed, users will have access to the additional data without having to reconnect.

This option is not applicable if the cube's structure has changed, or if the data from which the cube was created has changed.

Caution This option updates a partition. Incorrect use of partitions can result in inaccurate cube data. For more information, see Managing Partitions.

Refresh data

Select to cause the cube's data to be cleared and reloaded and aggregations recalculated. This option is applicable when the underlying data in the data warehouse has changed but the cube's structure remains the same.

A refresh can be performed while users continue to query the cube; after the refresh has completed, users will have access to the refreshed data without having to reconnect.

Full Process

Select to completely rebuild your cube after you have made structural changes to it, such as adding a dimension, dimension level, or measure. This option restructures the cube based on its current definition and recalculates and reloads its data. This option can take a long time.

After you process a cube with this option, users must reconnect to the server computer to continue working with the cube.

If a shared dimension's structure is updated and saved but not processed, it will automatically be processed when any cube incorporating the dimension is processed with this option. At that time any other cubes that incorporate the dimension immediately become unavailable to users and must be processed before they can be used again.

Incrementally update the dimensions of this cube

Select to enable incremental updating of the cube's shared and private dimensions whenever the cube is processed.

Settings

Click to display the **Cube Processing Settings Dialog Box**, where you can specify additional cube processing options.

See Also

Cube Processing Settings Dialog Box

Processing Cubes

Process a Dimension Dialog Box

Use this dialog box to process a shared dimension.

This dialog box appears when in the Analysis Manager tree pane, in the Shared Dimensions folder, you right-click a shared dimension, and then click **Process**.

Options

Incremental update

Select this option to append new dimension members to your shared dimension. This option is appropriate when the structure and relationship of the dimension levels and members have not changed, but new members (that is, rows) have been added to the dimension table.

A cube that incorporates a shared dimension remains available to users while the dimension is incrementally updated, and the added dimension members are available in the cube after the update is complete.

Rebuild the dimension structure

Select this option to completely rebuild the dimension structure. This option is appropriate after you have made changes to the structure or relationship of the dimension levels or members. For example, use this option after you add, delete, or move a level, or after you move a member from one parent member to another.

IMPORTANT If a shared dimension's structure is updated and saved but not processed, it will automatically be processed when any cube incorporating the dimension is processed. At that time any other cubes that incorporate the dimension immediately become unavailable to users and must be processed before they can be used again.

Process a Mining Model Dialog Box

Use this dialog box to choose how to process a relational or OLAP data mining model.

This dialog box appears when you right-click a relational or OLAP data mining model in the Analysis Manager tree pane, and then click **Process**.

Options

Refresh data

> Select to cause the content for the data mining model to be cleared and retrained.
>
> A refresh can be performed while users continue to access the data mining model; after the refresh has completed, users will have access to the refreshed data without having to reconnect.

Full process

> Select to completely rebuild your data mining model after you have made structural changes to it, such as adding a supporting table or a data mining column or changing the usage of an existing data mining column. This option restructures the data mining model based on its current definition, and fully trains the mining model. This option can take a long time.
>
> After you process a data mining model with this option, users must reconnect to the server computer to continue working with the data mining model.

See Also

Training Data Mining Models

Properties Dialog Box

Use this dialog box to specify options for warnings, file locations, performance and memory settings, processing, logging, add-ins, and Active Directory™.

This dialog box appears when you right-click a server in the Analysis Manager tree pane and then click **Properties**.

The **Properties** dialog box has six tabs. For more information about the tabs, click a link in the following table.

To do this	See
Specify whether certain warnings are displayed and to change the folders used by Microsoft® SQL Server™ 2000 Analysis Services.	General Tab (Properties Dialog Box)
Change performance and memory settings.	Environment Tab (Properties Dialog Box)
Change optimization settings.	Processing Tab (Properties Dialog Box)
Change the settings for the automatic logging of queries performed by Analysis Services and the automatic logging of processing messages. Options in this tab affect the output of the Usage-Based Optimization Wizard and the Usage Analysis Wizard.	Logging Tab (Properties Dialog Box)
Control which add-ins are started automatically with Analysis Manager.	Add-ins Tab (Properties Dialog Box)
Register an Analysis server with Active Directory, and set properties.	Active Directory Tab (Properties Dialog Box)

Options

The following option is displayed for all tabs.

Reset Defaults

Resets options on all tabs of the **Properties** dialog box, except the **Add-ins** tab, to their default installation values.

General Tab (Properties Dialog Box)

Use this tab to specify whether certain warnings are displayed and to change the folders used by Microsoft® SQL Server™ 2000 Analysis Services.

Options

Enable dimension level counting

Select to cause Analysis Services to count the members in a new dimension level when it is added. If you select this option, whenever a user attempts to add a level below a level with more members, the following warning is displayed.

Although this option provides real-time hierarchical structure validation, counting large numbers of members may take several minutes or longer. Turning off the option speeds up the dimension creation process; however, the hierarchical structure of members is not validated.

Default: Members are counted automatically, and warnings are displayed.

Note In the Dimension Wizard, you can temporarily override this option by using the **Count level members automatically** check box. The override is effective only for the duration of the wizard.

Show process warning

Select to cause Analysis Services to display the following warning when a new or updated cube is saved without aggregations.

Individual administrators can turn off the option by selecting the **Don't show this message again** check box in the **Design Storage** dialog box. Thereafter, the only way to begin displaying the warnings again is to select the **Show process warning** check box in the **Properties** dialog box.

Default: Warnings are displayed.

Show database password warning

Select to cause Analysis Services to display a warning when a data source is saved with a password. The following warning is displayed when in the **Data Link Properties** dialog box, on the **Connection** tab, an administrator specifies a user name and password, selects the **Allow saving of password** check box, and then clicks **OK**.

In the **Security Warning** dialog box, the administrator can click **Cancel** to avoid the security exposure.

Individual administrators can turn off the option by selecting the **Don't show this message again** check box in the **Security Warning** dialog box. Thereafter, the only way to begin displaying the warnings again is to select the **Show database password warning** check box in the **Properties** dialog box.

Default: Warnings are displayed.

To display warnings when data source passwords are saved, select the **Show database password warning** check box, and then click **OK**.

Data folder

Select the Data folder used by Analysis Services. If you select a new folder, the temporary file folder (see below) is also changed to that folder. If you select a new folder, Analysis Services does not access existing data.

IMPORTANT The Data folder stores security files that control end users' access to objects on the Analysis server. For this reason, you must secure the Data folder against unauthorized access.

The maximum allowed length is 102 characters.

Default: C:\Program Files\Microsoft Analysis Services\Data

To change the Data folder used by Analysis Services, click **Browse** next to the **Data folder** box, select a new folder, and then click **OK**.

Temporary file folder

Select the temporary folder used by Analysis Services. If you select a new Data folder, the temporary file folder is also changed to that folder.

The maximum allowed length is 102 characters.

Default: the folder specified in the **Data folder** box.

To change the temporary file folder used by Analysis Services, click **Browse** next to the **Temporary file folder** box, select a new folder, and then click **OK**. If you change this folder, you must stop and then restart the Analysis server service (MSSQLServerOLAPService).

Environment Tab (Properties Dialog Box)

Use this tab to change performance and memory settings.

Options

Maximum number of threads

Set the number of threads that can simultaneously use the CPUs in the server computer.

Default: two times the number of CPUs in the server computer.

Valid range: 1-1000.

To change the maximum number of threads, in the **Maximum number of threads** box, specify the maximum number. Only integers are valid.

Large level defined as

Set the minimum number of members a dimension level must contain to be processed as a large level.

Microsoft® SQL Server™ 2000 Analysis Services classifies dimension levels into large and small categories based on the number of level members. These two categories are processed differently. For example, large levels are not sent to the client unless they are specifically requested, but small levels are sent to the client even if the entire level is not requested.

Default: 1,000 members.

Valid range: 1-10,000 members.

To change the definition of large levels, in the **Large level defined as** box, specify the minimum number of members a dimension level must contain to be processed as a large level. Only integers are valid.

Minimum allocated memory

Set the amount of memory allocated exclusively to Analysis Services. Actual memory usage may exceed this value.

Default: one-half of the server computer's memory.

To change the amount of memory allocated to Analysis Services, in the **Minimum allocated memory** box, specify the allocated memory in megabytes (MB). Only integers are valid.

Memory conservation threshold

Set the memory threshold near which memory usage decreases. When the amount of memory used by Analysis Services approaches this threshold, usage decreases as memory management becomes more efficient. Actual memory usage may exceed this value unless the default value is used.

Default: all of server computer's memory.

To change the threshold for memory usage, in the **Memory conservation threshold** box, specify the threshold in MB. Only integers are valid.

Processing Tab (Properties Dialog Box)

Use this tab to change optimization settings.

Options

Read-ahead buffer size

Set the maximum amount of data placed into memory during each read of the database. Setting this option as high as possible minimizes the number of disk accesses.

Default: 4 megabytes (MB).

To change the read-ahead buffer size, in the **Read-ahead buffer size** box, specify the buffer size in MB. Only integers are valid.

Process buffer size

Set how much data is processed in memory before an I/O is performed. The larger the value, the fewer the I/Os.

Default: 4 MB.

To change the process buffer size, in the **Process buffer size** box, specify the buffer size in MB. Only integers are valid.

Server timeout

Set the maximum time Microsoft® SQL Server™ 2000 Analysis Services waits for a reply from a query to the data source when an object is processed. If the **Server timeout** check box is selected and the specified time elapses without a reply, a timeout occurs, and processing fails.

Default: 0 (disabled).

Valid range: 0 (disabled) - 1,000,000 seconds (approximately 11 days, 13 hours, 46 minutes).

To enable a server timeout, select the **Server timeout** check box. Then, in the **Seconds** box, specify the amount of time that elapses between submission of a query and timeout. Only integers are valid.

Logging Tab (Properties Dialog Box)

Use this tab to change the settings for the automatic logging of queries performed by Microsoft® SQL Server™ 2000 Analysis Services and the automatic logging of processing messages.

The Usage-Based Optimization Wizard and Usage Analysis Wizard rely on the query log. Options in this tab affect the output of these wizards.

Options

Log queries sent to server

Select to enable query logging.

- **Write to log once per __ queries**

  Specify the frequency of query logging. By default, every tenth query is logged. You can turn off query logging altogether or change the frequency.

  IMPORTANT Setting this option too low may adversely affect performance. Only integers are valid for this setting.

  Default: 10 (1 in 10 queries is logged).

  Valid range: 1-10,000.

Clear log

Select to delete the contents of the query log.

Consider clearing the query log after you run the Usage-Based Optimization Wizard consecutively on all cubes on the server.

To clear the query log, click **Clear log**, and then click **Yes**.

Log processing messages to a file

Select to enable the logging of processing messages to a log file (.log).

- **File path and name**

  Type a path and a name for the log file. UNC paths are recommended.

- **Browse**

  Click to display the **Open** dialog box, where you can select a path and specify a file name for the **File path and name** box.

See Also

Analyzing Usage Patterns

Optimizing Performance Based on Usage

Usage Analysis Wizard

Usage-Based Optimization Wizard

Add-ins Tab (Properties Dialog Box)

Use this tab to control which add-ins are started automatically with Analysis Manager. Add-ins are other applications that are compatible with Microsoft® SQL Server™ 2000 Analysis Services but may be produced by companies other than Microsoft. Changes on this tab do not take effect until Analysis Manager is restarted.

IMPORTANT Unlike the other tabs in the **Properties** dialog box, the **Add-ins** tab always applies to the local computer even if you displayed the dialog box by selecting a remote server.

Options

Available Add-ins

View the installed add-ins. Add-ins with selected check boxes are started automatically when you start Analysis Manager. They are started in the order displayed.

Priority

Set the order in which add-ins are started with Analysis Manager.

To change the order in which add-ins are started, click the add-in to change, and click one of the **Priority** buttons repeatedly until they are in the order you want.

Description

View a description of the add-in selected in the **Available Add-ins** box.

Reset Defaults

This button does not affect the options on this tab.

Active Directory Tab (Properties Dialog Box)

Use this tab to register an Analysis server with Active Directory™ directory services and provide information about the server. You can also use this tab to edit or delete the information you provide.

Options

Enable Active Directory registration

> Select this check box to enable the options on this dialog box and to register the server with Active Directory.

Keywords

> Define a comma-separated list of search keywords used to identify this server. For example, you can indicate the organization or workgroup, whether the server is dedicated to production or development activities, or the location of the server.

InformationURL

> Specify a URL to a Web page containing detailed information about the server or server object. The URL that you specify must correspond to a custom Web page that you create and publish within your intranet. The page that you create should contain additional information about the server.

Contact

> Specify the name of the person responsible for maintaining the Analysis server or the Microsoft® SQL Server™ 2000 instance that hosts the Analysis server. The values you enter are stored as a text string. You can type the contact's full name, an e-mail alias, or both.

Name

> Type a descriptive name that identifies the Active Directory server that contains registration information about the Analysis server. The name that you provide serves as a user-friendly alias that is intended as an alternative to

the computer name. This name is stored as a text string. It does not need to be unique. The name can include special characters or symbols, spaces, and alphanumeric characters.

See Also

Using Active Directory with Analysis Services

Relational Mining Model Editor

Use this tool to browse the content of a relational data mining model and edit its structure. You can also use Relational Mining Model Editor and associated dialog boxes to perform various procedures.

The following diagram shows the elements that make up Relational Mining Model Editor.

Unlike OLAP Mining Model Editor, Relational Mining Model Editor supplies two views, schema and content. Use the schema view to view the case and supporting tables that help in defining the data mining columns for the mining model. Use the content view to examine the data mining model content. You can switch from one view to another by clicking the **Schema** tab or the **Content** tab at the bottom of Relational Mining Model Editor. You can also click **Schema** or **Content** on the **View** menu.

Relational Mining Model Editor appears when you do one of the following:

- In the Analysis Manager tree pane, right-click a data mining model, and then click **Edit**.
- In the Analysis Manager tree pane, click a data mining model, and then on the **Action** menu, click **Edit**.

Relational Mining Model Editor has five tabs. For more information about the tabs, click a link in the following table.

To do this	See
Perform various basic tasks within Relational Mining Model Editor.	Menu Bar (Relational Mining Model Editor)
Carry out common functions within Relational Mining Model Editor.	Toolbar (Relational Mining Model Editor)
View the mining model structure, such as column types.	Structure Pane (Relational Mining Model Editor)

View and edit properties associated with the relational mining model and its data mining columns. View descriptions of properties and the objects to which they apply.	Properties Pane (Relational Mining Model Editor)
Display two views of the mining model provided by the **Data** and the **Schema** tabs in the browser pane.	Browser Pane (Relational Mining Model Editor)

Menu Bar (Relational Mining Model Editor)

Use the menu bar to perform various tasks with relational data mining models.

The following table describes the menus and their associated options.

Menu item	Description
File	Saves relational data mining models and exits Relational Mining Model Editor.
Edit	Allows you to rename or delete columns in the relational data mining model, as well as removing or reversing joins in the schema of the mining model.
View	Provides schema, content and property viewing options.
Insert	Inserts tables, columns, and nested tables into the relational data mining model.
Tools	Supports processing of the relational data mining model.
Help	Provides access to Help about Relational Mining Model Editor, as well as access to SQL Server Books Online, which contains all of the Microsoft® SQL Server™ 2000 Analysis Services documentation.

Toolbar (Relational Mining Model Editor)

Use the toolbar to access the most commonly-used functions in Relational Mining Model Editor.

The following table describes the toolbar buttons in greater detail.

Button	Description
Save	Saves the relational data mining model.
Insert Table	Adds a new table to the schema of the relational data mining model.
Insert Column	Adds a new column to the structure of the relational data mining model.
Insert Nested Table	Adds a new nested table to the structure of the relational data mining model.
Process Mining Model	Displays the **Process a mining model** dialog box, where you can select the processing method for the relational mining model.

Structure Pane (Relational Mining Model Editor)

The structure pane provides a graphical representation of the usage of each data mining model column in the mining model, as shown in the following diagram.

The following table describes column types in more detail.

Column type	Description
Key Column	The column the mining model uses as a key column. The key column is an identifying column for the case set.
Input Column	The column the mining model uses as an attribute column for input for the case set.
Predictable Column	The column the mining model uses as a predictive column for output for the case set.
Input and Predictable Column	The column the mining model uses as an attribute column for both input and output for the case set.

Properties Pane (Relational Mining Model Editor)

Use the properties pane in Relational Mining Model Editor to view properties associated with the relational data mining model and its data mining model columns.

The following table describes the features of the properties pane.

Feature	Description
Properties button	Shows or hides the properties pane.
Basic tab	Shows the most commonly used properties, such as **Name** and **Description**, for the mining model and mining model columns.
Advanced tab	Displays advanced properties, such as **Distribution** and **Content Type**, used to further define the mining model columns.
Description	Displays the name and a brief explanation of the property selected in the properties pane.

Basic Properties

The **Basic** tab is used to display and, optionally, edit the most commonly viewed properties for data mining models and data mining columns. The following table describes the properties displayed in the **Basic** tab in more detail, and it indicates the data mining object (data mining model or data mining column) to which the property applies.

Property	Description	Applicable objects
Name	The name of the selected data mining model or column. This property is read-only for data mining models.	Both

Description	The description of the selected data mining model or column.	Both
Mining Algorithm	The data mining algorithm provider for the selected data mining model.	Data mining model
Are Keys Unique	Whether the key columns in the data mining model uniquely identify records in the source case table.	Data mining model
Is Case Key	Whether the data mining column is used as a key column in the data mining model. This property must be set to **False** before you can delete the column.	Data mining column
IsNestedKey	Whether the data mining column is used as a key column for a nested table in the data mining model. This property must be set to **False** before you can delete the column.	Data mining nested table column
Source Column	The name of the source column in the case or supporting table.	Data mining column
Data Type	The data type of the data mining column. This setting must be compatible with the data mining algorithm provider that is being used. The data types and algorithms that Microsoft® SQL Server™ 2000 Analysis Services supports are documented in the OLE DB for Data Mining specification. For more information about the OLE DB for Data Mining specification, see the Microsoft OLE DB Web page at the Microsoft Web site. For data types supported by data mining algorithm providers, see the data mining algorithm provider	Data mining column

	documentation.	
Usage	Whether the data mining column is used as an input column, a predictable column, or both. This property is read-only for key columns.	Data mining column
Additional Parameters	A comma-delimited list of provider-specific mining parameter names and values. For mining parameters supported by data mining algorithm providers, see the data mining algorithm provider documentation.	Data mining model

Advanced Properties

The **Advanced** tab displays advanced properties for data mining models and data mining columns, such as relation column information and distribution. The following table describes these advanced properties in more detail.

Property	Description	Applicable objects
Related To	For relation columns, the name of the column to which the selected data mining column is related. It is read-only for key columns. When this property is set for a column, the **Usage** property is changed to match the value of the related column.	Data mining column
Distribution	The distribution flag, such as NORMAL or UNIFORM, of the data mining column. It is read-only for key columns.	Data mining column
Content Type	The content type, such as DISCRETE or ORDERED, for the data mining column. It is read-only	Data mining column

	for key columns.	
Data Options	The model flag, such as MODEL_EXISTENCE_ONLY or NOT NULL, of the data mining column. It is read-only for key columns.	Data mining column

Browser Pane (Relational Mining Model Editor)

The browser pane provides two views of the relational data mining model. The **Schema** tab shows the case and associated tables used to construct the relational data mining model. The **Content** tab shows the mining model content for processed relational data mining models. The **Content** tab uses Data Mining Model Browser to illustrate the mining model content.

The following diagram shows the **Schema** tab with a displayed case table.

The following diagram shows the **Content** tab with a displayed Microsoft® Decision Trees node structure.

Register Analysis Server Dialog Box

Use this dialog box to register Analysis servers.

To register an Analysis server, your user name must be included in the OLAP Administrators group on that Analysis server. If your user name is added to the OLAP Administrators group, you must log off and log on Microsoft® Windows® before you can register the Analysis server.

This dialog box appears when you:

- Right-click the Analysis Servers folder in the Analysis Manager tree pane and then click **Register Server**.

  -or-

- Click **Register Server** in the Partition Wizard.

Options

Server name

Type the name of the Analysis server to which you want to connect.

Note You cannot register an Analysis server by specifying its IP address.

Register Function Libraries Dialog Box

Use this dialog box to register function libraries of other products or user-defined function libraries.

You can create user-defined functions to provide functions not in the Microsoft® SQL Server™ 2000 Analysis Services function library. A user-defined function can be developed on any tool capable of generating Microsoft ActiveX® libraries.

You can register user-defined function libraries of the following types:

- Type libraries (*.olb, *.tlb, *.dll)
- Executable files (*.exe, *.dll)
- ActiveX controls (*.ocx)

The Analysis Services functions always take precedence over registered function libraries when function name conflicts occur (that is, two libraries contain the same function name). Except Analysis Services functions, the order of the libraries in the list is important. It defines the precedence of function resolution in cases of name conflicts. The first library takes precedence.

Analysis Services automatically registers the Microsoft Visual Basic® for Applications Expression Services function library (VBA332.dll) and the library that supports the Microsoft Excel worksheet functions. However, the Excel library must be installed separately from Analysis Services. For more information, including lists of supported functions, see Visual Basic for Applications Functions and Excel Functions.

This dialog box appears when you click **Register** in Calculated Member Builder or MDX Builder. After libraries are registered, the program IDs of the libraries appear in the **Functions** box of Calculated Member Builder or MDX Builder. You can return to the **Register Function Libraries** dialog box and add or remove other libraries.

Note Functions are supported only if they accept as arguments only string or numeric data types, or array or variant data types containing string or numeric values. In addition, functions are supported only if they return only string or numeric data types, or variant data types containing numeric values.

Options

Add

Click to locate and select the library file you want to register. The program IDs associated with the function library appear in the list box.

Remove

Click to remove a selected program ID from the selection to be registered.

Close

Click to complete registration when you are satisfied that the correct programs in external function libraries are contained in the list box.

Calling a User-Defined Function within MDX

After a user-defined function is registered in the **Register Function Libraries** dialog box, it can be used anywhere in the Multidimensional Expressions (MDX) syntax that allows expressions, for example:

```
With Member Measures.[Forecasted Sales] As
   'Sales * ForecastedGrowthRate(SaleReps.CurrentMember.Name)'
Select TopCount(SalesReps, HowManyReps(), Sales) on Rows,
   {Sales, [Forecasted Sales] } on Columns
From Sales
```

The HowManyReps and ForecastedGrowthRate user-defined functions are defined as:

```
Public Function HowManyReps() as Integer
Public Function ForecastedGrowthRate(RepName as String) as Doubl
```

Note There may be multiple user-defined functions residing in the same

ActiveX library.

Remote Server Data Directory Dialog Box

Use this dialog box to confirm or change the path of the Data directory to which a Microsoft® SQL Server™ 2000 Analysis Services database will be restored.

For more information about restoring Analysis Services databases, see Archiving and Restoring Databases.

This dialog box is displayed when, in the Analysis Manager tree pane, you right-click a remote server and then click **Restore Database**.

Options

Path

Specify the path of the Data directory that will contain the files for the database after the restoration is complete.

The Data directory is created during installation of Analysis Services and contains a directory for each database on the Analysis server. The default path of the Data directory is:

```
C:\Program Files\Microsoft Analysis Services\Data\
```

You can change the \Program Files\Microsoft Analysis Services\ portion of the path. (To determine the current path, right-click the remote server, click **Properties**, and then see the **Data folder** box.)

In the **Path** box, you must precede the path of the Data directory with the remote server name. If the Data directory is shared at its disk level, you must also include the share name associated with the disk. The following example **Path** value includes the server name Server-1 and the disk share name C$.

```
\\Server-1\C$\Program Files\Microsoft Analysis Services\Data\
```

Browse

Click to display a dialog box in which you can browse available paths and select one.

OK

Click to select the path in the **Path** box and continues the restoration procedure.

If **OK** is unavailable (dimmed), the path in the **Path** box is not valid or you do not have authorization to access it.

Restore Database Dialog Box

Use this dialog box to begin the restoration of a Microsoft® SQL Server™ 2000 Analysis Services database.

Caution When you restore a database, its file set (in the Data directory and its subdirectories) and its meta data are returned to their states at the time the archive file was created. Files that were created since then in these directories are deleted. Changes and additions since then to Analysis Services repository records for the database and its objects are removed.

For more information about restoring Analysis Services databases, see Archiving and Restoring Databases.

This dialog box is displayed when in the **Open Archive File** dialog box you click **Open**. To open the **Open Archive File** dialog box, right-click the server on which you want to restore the database, and then click **Restore Database**.

Options

Restore

Click to start the restoration and display the **Restore Database Progress** dialog box, where you can monitor or cancel the restoration.

Restore Database Progress Dialog Box

Use this dialog box to monitor the restoration of a Microsoft® SQL Server™ 2000 Analysis Services database. You can also cancel the restoration or save the restore log.

For more information about restoring Analysis Services databases, see Archiving and Restoring Databases.

This dialog box is displayed when, in the **Restore Database** dialog box, you click **Restore**. To restore a database, right-click the server on which you want to restore the database, and then click **Restore Database**. This displays the **Open Archive File** dialog box. Select an archive to restore, and then click **Open** to display the **Restore Database** dialog box.

Options

Save Log

Click to display a dialog box in which you specify the path and file name of the restore log and then save it. If the restoration completes successfully, the restore log contains a list of the restored files. Otherwise, the restore log contains one or more messages. **Save Log** is available only after the restoration is completed or canceled.

Cancel

Click to cancel the restoration. If you click **Cancel** and receive a message indicating that the extraction was cancelled, nothing has been restored. A delay may occur between clicking **Cancel** and display of the message.

Cancel is replaced by **Close** after the restoration is completed or canceled.

Close

Click to close the dialog box. **Close** is available only after the restoration is completed or canceled.

Save Object Dialog Box

Use this dialog box to name objects.

This dialog box appears at various times when you save new or edited objects.

Option

Object name

Type a name for the object. For a dimension name, you can enter a maximum of 24 characters. For the names of other objects, such as cubes, you can enter a maximum of 50 characters.

Select Column Dialog Box

Use this dialog box to select a column in the Dimension Wizard, Dimension Editor, Cube Editor, or Relational Mining Model Editor.

This dialog box appears when you perform one of the following actions:

- In the Dimension Wizard, specify ordering options for a new dimension.

- In Dimension Editor or Cube Editor, specify various property values.

- On the Relational Mining Model Editor toolbar, select **Insert Column**.

Options

Tree pane

View the available columns.

Column Types

Select or clear check boxes to display columns of the type described.

Select Nested Table Key Column Dialog Box

Use this dialog box to specify the key column for a nested table in a relational data mining model. To add a nested table to the mining model definition, you must first add it as a supporting table in the **Schema** tab of Relational Mining Model Editor.

This dialog box appears when you do one of the following in Relational Mining Model Editor:

- Click the **Insert Nested Table** button on the toolbar.
- Click **Nested Table** from the **Insert** menu.

Options

Select a column

Choose a column from an available supporting table to become the key for the new nested table.

Column Types

Select a data type to restrict the selection of available columns.

OK

Click to add the nested table to the mining model and to close the dialog box.

See Also

Relational Mining Model Editor

Select Table Dialog Box

Use this dialog box to select a table from a selected data source.

This dialog box appears at various times when you work in Cube Editor, Dimension Editor, or Relational Mining Model Editor.

Options

Tables

View the tables of the selected data source.

Details

View a description of the data source if a data source is selected in the **Tables** box. View the columns in the table if a table is selected in the **Tables** box.

New Data Source

Click to display the **Data Link Properties** dialog box, where you can specify a new data source. This button is unavailable for some operations.

Add

Click to select the table you choose. If you are replacing a table, click **OK**.

Select the Parent Member Dialog Box

Use this dialog box to select a different parent member for your calculated member. The parent member determines the location of the calculated member in the dimension structure.

This dialog box appears when you click **Change** in Calculated Member Builder.

Options

Tree box

Expand the dimension and select the parent member you want for the calculated member.

Select Users Dialog Box

Use this dialog box to:

- Select user names and groups by whose query usage you want to optimize. Display this dialog box from the Usage-Based Optimization Wizard.

- Select user names and groups whose query usage you want to analyze. Display this dialog box from the Usage Analysis Wizard.

Groups and users in roles granted access to the cube are displayed in the **Available users** box.

To select a group or user, click the group or user in the **Available users** box, and then click **Add**.

The **Select Users** dialog box appears when in either wizard you select the **Queries by these users** check box, and then click **Add**.

Options

Available users

View the groups and users in roles granted access to the cube.

Add

Click to add the selected group or user to the **Selected users** box.

Selected users

View the groups and users you select.

OK

Click to return to the wizard and place the groups and users you selected in the **Queries by these users** box.

Set Default Member Dialog Box

Use this dialog box to:

- Clear a previously specified custom default member.

- Specify a default member by selecting it in a member tree or by creating an expression in Multidimensional Expressions (MDX).

The default member of a dimension slices the datasets returned by queries when the dimension is not displayed on an axis and no slicing member in the dimension is specified.

Note The default member can be overridden in individual roles. For more information, see Custom Rules in Dimension Security.

This dialog box appears when, in the tree pane of Dimension Editor or Cube Editor, you select a dimension, click **Default Member** in the properties pane, and then click the edit (**...**) button.

Options

No custom default

Select to clear the specification of a default member.

If the **All Level** property of the dimension is set to **Yes**, the member indicated by the **All Caption** property will be the default member. If the **All Level** property of the dimension is set to **No**, an arbitrary member of the highest level will be the default member.

Choose member to be the default

Expand the member tree and select a default member.

Enter MDX formula to specify default member for dimension

Type an MDX expression to represent the default member. For example, to specify Paris as the default member of the Location dimension, type:

[Location].[All Location].[Europe].[France].[Paris]

You can also click the expand (**...**) button to display MDX Builder, where you can construct an MDX expression.

OK

Click to temporarily save your changes and close the dialog box. To permanently save your changes, in the editor, on the **File** menu, click **Save**.

Training Query Dialog Box

Use this dialog box to modify a training query used to process a mining model within a Data Transformation Services (DTS) Analysis Services processing task.

This dialog box is displayed when, in the Analysis Services Processing Task dialog box tree pane, you select a relational mining model and click the edit (**...**) button next to the **Training query** box.

Options

Training query

Type the Multidimensional Expressions (MDX) syntax for the training query. Syntax must conform to the OLE DB for Data Mining specification. For more information about the OLE DB for Data Mining specification, see the Microsoft® OLE DB Web page at the Microsoft Web site.

See Also

Analysis Services Processing Task Dialog Box

MDX

Virtual Cube Editor

Use this tool to browse a virtual cube's data and to examine and edit the structure of a virtual cube. In addition, with Virtual Cube Editor and associated dialog boxes, you can perform various procedures.

Virtual Cube Editor appears when you do either of the following:

- In the Analysis Manager tree pane, right-click a virtual cube, and then click **Edit**.

- In the Analysis Manager tree pane, click a virtual cube, and then on the **Action** menu, click **Edit**.

- Virtual Cube Editor has five areas. For more information about the areas, click a link in the following table.

To do this	See
Perform commands available in the Virtual Cube Editor menus.	Menus (Virtual Cube Editor)
Perform common actions represented by icons on the Virtual Cube Editor toolbar.	Toolbar (Virtual Cube Editor)
View objects of the virtual cube displayed in the tree pane.	Tree Pane (Virtual Cube Editor)
View and modify certain properties of the object selected in the Virtual Cube Editor tree pane. View descriptions of properties and the objects to which they apply.	Properties Pane (Virtual Cube Editor)
View the cube data in a table	Data Pane (Virtual Cube Editor)

format. Exchange or move dimensions. Slice through a dimension, or drill down into a member.	

See Also

Building a Virtual Cube

Creating and Maintaining Calculated Members in Virtual Cubes

Specifying Drillthrough Options

Menus (Virtual Cube Editor)

Use the following menu options to perform operations in Virtual Cube Editor.

Menu	Option	Description
File	**Save**	Saves the virtual cube.
	Save As	Saves the virtual cube under a different name.
	Exit	Closes Virtual Cube Editor.
Edit	**Structure**	Displays the Virtual Cube Wizard so you can modify the virtual cube's structure.
	Edit	Edits the selected object. This option is unavailable for objects that have no associated editor or wizard.
	Rename	Renames the selected object, if applicable.
	Delete	Deletes the selected object, if applicable.
	Import	Displays options for the types of objects to be imported into the virtual cube. Options include calculated members, actions, or named sets.
View	**Properties**	Expands or collapses the properties pane.
	Browsing Enabled	Retrieves and displays cube data for browsing. Until this option is reset, it retains the previous setting.
Insert	**Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
	Calculated Cells	Displays Calculated Cells wizard so you can begin creating calculated cells.
	Action	Displays the Action Wizard so you can create a new action.
	Named Set	Displays Named Set Builder so you can create a new named set.
Tools	**Process Virtual**	Processes the virtual cube.

	Cube	
Help	**Help on Virtual Cube Editor**	Displays a Help topic about Virtual Cube Editor.
	SQL Services Books Online	Opens SQL Server Books Online.

Toolbar (Virtual Cube Editor)

Use the following toolbar buttons to perform common operations.

Button	Description
Save	Saves the cube.
Edit Structure (Wizard)	Starts the Virtual Cube Wizard so you can modify the structure of the virtual cube.
Insert Calculated Member	Displays Calculated Member Builder so you can begin creating calculated members.
Insert Calculated Cells	Displays the Calculated Cells Wizard so you can begin creating calculated cells.
Insert Action	Displays the Action Wizard so you can create a new action.
Insert Named Set	Displays Named Set Builder so you can create a new named set.
Import Calculated Members	Displays the **Import Calculated Members** dialog box so you can import calculated members from a source cube or from another cube.
Import Calculated Cells	Displays the **Import Calculated Cells** dialog box so you can import calculated cells from a source cube or another cube.
Import Actions	Displays the **Import Actions** dialog box so you can import actions from a source cube or from another cube.
Import Named Sets	Displays the **Named Sets** dialog box so you can import named sets from a source cube or from another cube.

Tree Pane (Virtual Cube Editor)

Use the tree pane to display objects in the cube. Right-click an object to see a shortcut menu for that object.

The following table lists how to access available information about cube objects.

Right-click	Shortcut menu option	Description
Virtual cube	**Process Virtual Cube**	Processes the virtual cube.
	Edit Structure (Wizard)	Displays the Virtual Cube Wizard so you can modify the virtual cube's structure.
Dimensions folder	**Edit Structure (Wizard)**	Displays the Virtual Cube Wizard so you can modify the virtual cube's structure.
Dimension	**Edit Structure (Wizard)**	Displays the Virtual Cube Wizard so you can modify the virtual cube's structure.
Level	**None**	--
Measures folder	**Edit Structure (Wizard)**	Displays the Virtual Cube Wizard so you can modify the virtual cube's structure.
Measure	**Edit Structure (Wizard)**	Displays the Virtual Cube Wizard so you can modify the virtual cube's structure.
	Rename	Renames the measure.
Calculated members folder	**New Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
	Import Calculated Member	Displays the **Import Calculated Members** dialog box so you can import calculated members from a source cube

		or from another cube.
Calculated member	**New Calculated Member**	Displays Calculated Member Builder so you can begin creating calculated members.
	Import Calculated Member	Displays the **Import Calculated Members** dialog box so you can import calculated members from a source cube or from another cube.
	Edit	Displays the calculated member in Calculated Member Builder so you can edit it.
	Delete	Deletes the calculated member.
	Rename	Renames the calculated member.
Calculated Cells folder	**New Calculated Cells**	Displays the Calculated Cells Wizard so you can begin creating calculated cells.
	Import Calculated Cells	Displays the **Import Calculated Cells dialog box so you can import calculated** cells from a source cube or another cube.
Calculated cells	**New Calculated Cells**	Displays the Calculated Cells wizard so you can begin creating calculated cells.
	Import Calculated Cells	Displays the **Import Calculated Cells** dialog box so you can import calculated cells from a source cube or another cube.
	Edit	Displays the Calculated Cells Wizard so you can edit the selected calculated cells definition.
	Rename	Renames the selected calculated cells definition.
	Delete	Deletes the selected calculated cells action.
Actions folder	**New Action**	Displays the Action Wizard so you can create a new action.
	Import Action	Displays the **Import Actions** dialog box

		so you can import actions from a source cube or from another cube.
Action	**New Action**	Displays the Action Wizard so you can create a new action.
	Import Action	Displays the **Import Actions** dialog box so you can import actions from a source cube or from another cube.
	Edit	Displays the Action Wizard so you can edit the selected action.
	Rename	Renames the action.
	Delete	Deletes the action from the virtual cube.
Named sets folder	**New Named Set**	Displays Named Set Builder so you can create a new named set.
	Import Named Set	Displays the **Import Named Sets** dialog box so you can import named sets from a source cube or from another cube.
Named set	**New Named Set**	Displays Named Set Builder so you can create a new named set.
	Import Named Set	Displays the **Import Named Sets** dialog box so you can import named sets from a source cube or from another cube.
	Edit	Displays Named Set Builder so you can edit the selected named set.
	Rename	Renames the named set.
	Delete	Deletes the named set from the virtual cube.

Properties Pane (Virtual Cube Editor)

Use the properties pane to view the properties of the object selected in the tree pane. To display the properties pane, click **Properties** beneath the tree pane or click **Properties** from the **View** menu.

Each type of object contains a different set of properties. Use the properties pane to modify the property settings for the selected object. For shared dimensions, their levels, and their member properties, some properties are read-only and must be changed in Dimension Editor or by editing their source cubes in Cube Editor.

The following table describes the properties displayed in the properties pane.

Object	Property	Description
Action	**Action Type**	Indicates the kind of operation performed by the action.
	Application	Stores additional information required by the application used to perform the action.
	Caption Expression	The text displayed for an action if supported by the application used to perform the action. This is in the form of a Multidimensional Expressions (MDX) expression.
	Description	Displays the text string used to describe an action.
	Invocation	Determines how the action is activated. • **Interactive**: Actions run interactively when the end user activates an associated element in the user interface. This is the default setting and the only type supported by Cube Browser in Microsoft® SQL Server™ 2000 Analysis Services.

		• **On Open**: Actions run automatically when the client opens a virtual cube. • **Batch**: Actions are run by the client application in a batch job. These actions provide a way to check for exceptions.
	Name	Contains the name of the action. This property is read-only.
	Value	Displays the syntax for the action. The syntax is in the form of an MDX expression and can include literal expressions enclosed by quotation marks.
Calculated member	**BackColor**	Indicates the background color of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Virtual Cube Browser and the Virtual Cube Editor data pane do not support this property.
	FontFlags	Indicates the font flags of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Virtual Cube Browser and the Virtual Cube Editor data pane do not support this property.
	FontName	Indicates the font of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Virtual Cube Browser and the Virtual Cube Editor data pane do not support this property.
	FontSize	Indicates the font size of the displayed

		calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Virtual Cube Browser and the Virtual Cube Editor data pane do not support this property.
	ForeColor	Indicates the foreground color of the displayed calculated member. To use this optional property, the client application must support its use and interpret its values. Only numeric values are valid. Virtual Cube Browser and the Virtual Cube Editor data pane do not support this property.
	Format String	Indicates the format for displaying cell values. A list of common formats is displayed when you click **Format String** and display the drop-down list. This property accepts the same values as the **Display Format** property of measures. For more information, see Display Formats.
	Name	Contains the name of calculated member.
	Non Empty Behavior	Stores the name of the measure used to resolve NON EMPTY queries in MDX. If the **Non Empty Behavior** property is blank, the calculated member must be evaluated repeatedly to determine if a member is empty. If the **Non Empty Behavior** property contains the name of a measure, the calculated member is treated as empty if the specified measure is empty.
	Parent Dimension	Lists the dimension that includes the calculated member. If the calculated member is a measure, specify **Measures**.
	Parent Member	Lists the member that includes the calculated member. The parent member determines the location of the calculated member in the

		dimension structure. This property must be null if the parent dimension is a one-level dimension or **Measures**.
	Solve Order	Indicates the order in which the calculated member is resolved in case of intersection with other calculated members. Valid values are 0 (zero) and positive or negative integers. Calculated members with lower **Solve Order** values have precedence in resolution.
	Value	Contains an MDX expression that defines the values of the calculated member. The values are displayed to end users as they browse the virtual cube.
	Visible	Determines whether the calculated member is visible in the cube or not.
Calculated cells	**BackColor**	Indicates the background color of the cells defined by the cell coordinates and condition.
	Calculation Condition	Contains an MDX conditional expression that defines the calculation subcube cells that are computed with the calculation formula by testing each cell.
	Calculation Subcube	Contains an MDX set expression that defines the subset of the cube data that is computed using the calculation formula.
	Calculation Value	Contains an MDX expression that provides the value for each cell in the calculation subcube.
	Description	Contains the text string that describes the calculated cells definition.
	Disabled	Determines whether the cell calculation is disabled. The default value is **False**.
	FontName	Indicates the font name of the cells defined by the cell coordinates and condition.
	FontSize	Indicates the font size of the cells defined by the cell coordinates and condition.

	FontFlags	Indicates the font flags of the cells defined by the cell coordinates and condition.
	Calculation Pass Number	Indicates the calculation pass in which this calculation is executed.
	Calculation Pass Depth	Determines how many calculation passes are required to fully compute a calculated cells definition.
	ForeColor	Indicates the foreground color of the cells defined by the cell coordinates and condition.
	Format String	Contains the format string of the cells defined by the cell coordinates and condition.
	Name	Contains the name of the calculated cells definition. This property is read-only.
	Solve Order	Contains a number representing the order of evaluation of the calculated cells.
	Visible	Determines whether the calculated cells definition is visible in the schema rowset. The default value is **True**.
Cube	**Default Measure**	Indicates the measure that is returned by queries when no measure is displayed on an axis and no slicing measure is specified. If no default measure is specified, an arbitrary measure is the default measure.
	Description	Contains the text string used to describe the virtual cube.
	Enable Drillthrough	Indicates whether drillthrough is enabled for the virtual cube. For more information, see Specifying Drillthrough Options.
	Name	Contains the name of the virtual cube.
	Visible	Indicates whether the virtual cube is visible to end users as they browse the list of available cubes.
Dimension	**All Member Formula**	Contains the custom rollup formula for the (All) level. This formula is an MDX expression that determines the All member's

		cell values and overrides the **Aggregate Function** properties of measures. For more information, see Custom Rollup Formulas and Custom Member Formulas.
	Description	Contains the text string used to describe the dimension. This property is read-only in Virtual Cube Editor.
	Name	Contains the name of the dimension. This property is read-only in Virtual Cube Editor.
	Visible	Indicates whether the dimension is visible to end users as they browse the virtual cube.
Level	**Custom Rollup Formula**	Contains an MDX expression that overrides the default rollup of values in the level. For more information, see Custom Rollup Formulas and Custom Member Formulas.
	Description	Contains the text string used to describe the level. This is read-only in Virtual Cube Editor.
	Disabled	Indicates whether the level is available in the cube. This property is not available for parent-child dimensions. A level whose **Disabled** property is **Yes** cannot be referenced explicitly in calculated members and other MDX expressions.
	Name	Contains the name of the level. This property is read-only in Virtual Cube Editor.
	Visible	Indicates whether the level is visible to end users as they browse the cube. To set this property to **False**, you must first set the **Member Keys Unique** property of all lower levels to **True**. The **Visible** property is not available for parent-child dimensions. Unlike other objects whose **Visible** property is **False**, a level cannot be explicitly referenced in calculated members and other MDX expressions.

Measure	**Aggregate Function**	Contains the function used to aggregate measure values. This is read-only in Virtual Cube Editor.
	Data Type	Indicates the data type of the columns that store measure values in aggregations. This is read-only in Virtual Cube Editor.
	Description	Contains the text string used to describe the measure.
	Display Format	Indicates the format of the measure values displayed to end users as they browse the virtual cube. For more information, see Display Formats. This is read-only in Virtual Cube Editor.
	Name	Contains the name of the measure. The name is displayed to end users as they browse the virtual cube.
	Source Cube	Lists the source cube from which the selected measure's values are obtained. This is read-only in Virtual Cube Editor.
	Source Measure	Lists the source measure from which the selected measure's values are obtained. This is read-only in Virtual Cube Editor.
	Visible	Indicates whether the measure is visible to end users as they browse the virtual cube. If this value is **False** for a measure in a source cube, this property is read-only in Virtual Cube Editor.
Member property	**Caption**	Contains the caption used to display the member property. This is read-only in Virtual Cube Editor.
	Data Size	Indicates the maximum number of characters allowed in the column that stores the member property values. This is read-only in Virtual Cube Editor.
	Data Type	Indicates the data type of the column that stores the member property values. This is

		read-only in Virtual Cube Editor.
	Description	Contains the text string used to describe the member property. This is read-only in Virtual Cube Editor.
	Language	Identifies the client language of the member property. This is read-only in Virtual Cube Editor.
	Name	Contains the name of the member property. This is read-only in Virtual Cube Editor.
	Source Column	Contains the column in one of the dimension tables for the dimension that contains the member property that stores the values of the member property. This is read-only in Virtual Cube Editor.
	Type	Indicates to client applications of the type of information in the member property values. This is read-only in Virtual Cube Editor.
	Visible	Indicates whether the member property is visible to end users as they browse the virtual cube. This is read-only in Virtual Cube Editor.
Named set	**Name**	Contains the name of the named set.
	Value	Contains the MDX expression of the named set.

Data Pane (Virtual Cube Editor)

Use the data pane to display the virtual cube's data. The default view in the data viewing pane shows the data in table format with one dimension across the column headings and another dimension down the left column. (Measures are treated as a single dimension for this purpose.) The remaining dimensions of the virtual cube are displayed in the data slicing pane.

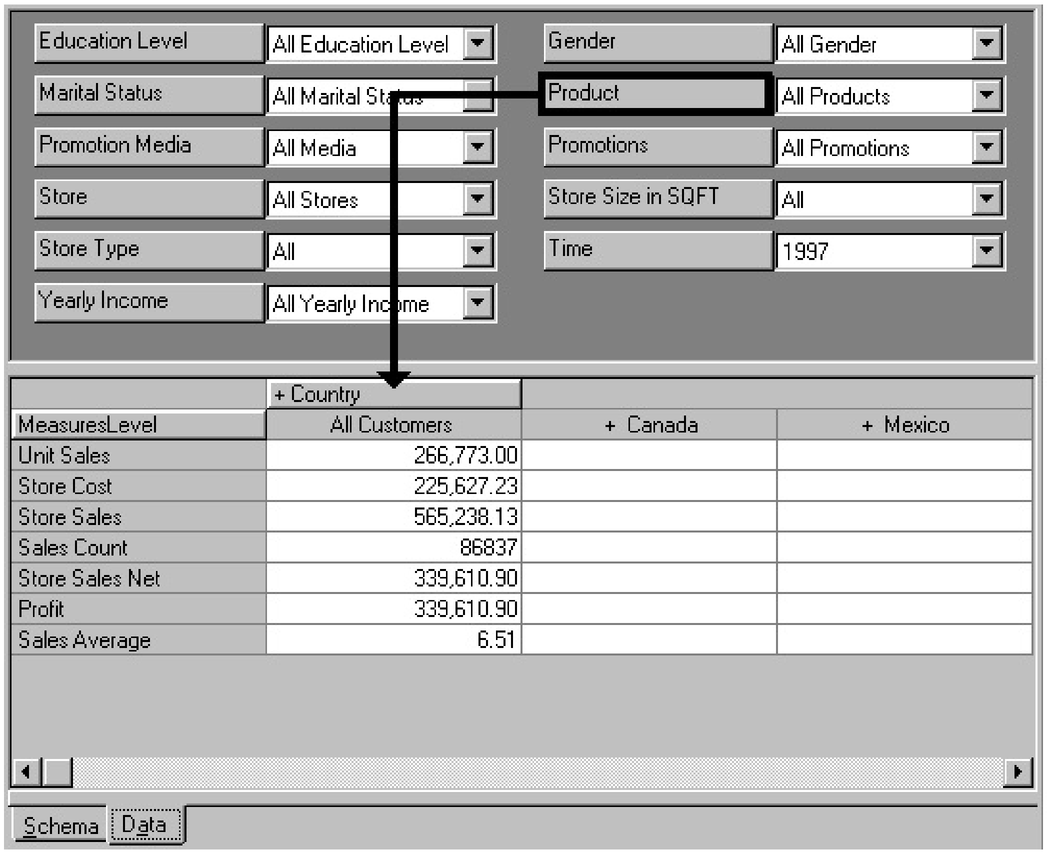
The white cells in the data viewing pane represent the measure values according to the members that appear in the members boxes for all the dimensions shown in the data slicing pane. In the preceding example, the measure values reflect the All member of every dimensions in the data slicing pane except Time. The members box for the Time dimension displays 1997, which indicates that the displayed measure values are limited to 1997. Later in this topic, an example shows you how to slice through dimensions in the data slicing pane to limit the measure values that are displayed. To browse the actual data, you can drill down or slice through your data.

Working in the Data Viewing Pane

To browse the cube data in the data viewing pane by any combination of dimensions, drag measures or dimensions from the data slicing pane onto either axis in the data viewing pane.

To replace a dimension in the data viewing pane with another dimension

- Drag the dimension from the data slicing pane to the dimension in the data viewing pane that you want to replace.



Education Level	All Education Level	Gender	All Gender
Marital Status	All Marital Status	Product	All Products
Promotion Media	All Media	Promotions	All Promotions
Store	All Stores	Store Size in SQFT	All
Store Type	All	Time	1997
Yearly Income	All Yearly Income		

	+ Country		
MeasuresLevel	All Customers	+ Canada	+ Mexico
Unit Sales	266,773.00		
Store Cost	225,627.23		
Store Sales	565,238.13		
Sales Count	86837		
Store Sales Net	339,610.90		
Profit	339,610.90		
Sales Average	6.51		

The pointer is in the correct position when it appears like this.

Exchange a slicer dimension with a row dimension.

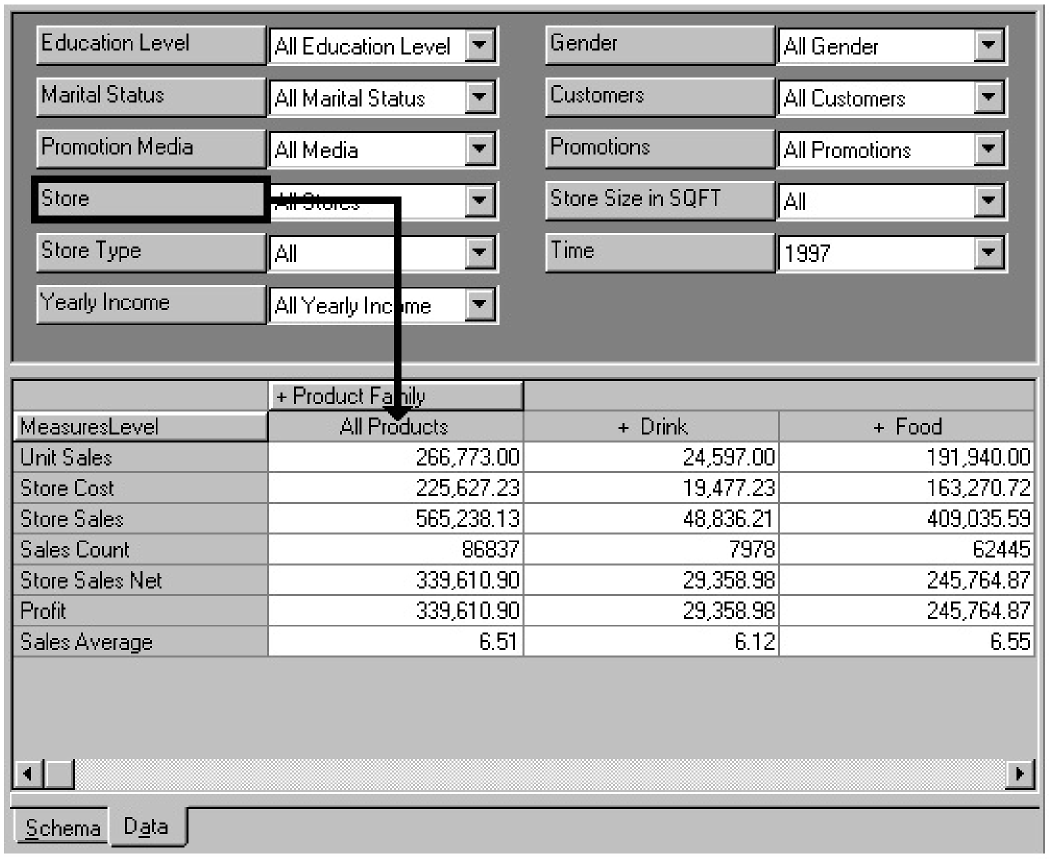
-or-

Exchange a slicer dimension with a column dimension.

The dimension in the data viewing pane moves up to the data slicing pane and the new one takes its place.

To move a dimension to the data viewing pane

- Drag the dimension from the data slicing pane to a cell below (or beside) the existing dimension.



Education Level	All Education Level	Gender	All Gender
Marital Status	All Marital Status	Customers	All Customers
Promotion Media	All Media	Promotions	All Promotions
Store	All Stores	Store Size in SQFT	All
Store Type	All	Time	1997
Yearly Income	All Yearly Income		

	+ Product Family		
MeasuresLevel	All Products	+ Drink	+ Food
Unit Sales	266,773.00	24,597.00	191,940.00
Store Cost	225,627.23	19,477.23	163,270.72
Store Sales	565,238.13	48,836.21	409,035.59
Sales Count	86837	7978	62445
Store Sales Net	339,610.90	29,358.98	245,764.87
Profit	339,610.90	29,358.98	245,764.87
Sales Average	6.51	6.12	6.55

The pointer is in the correct position when it appears like this.

Move a slicing dimension to a row.

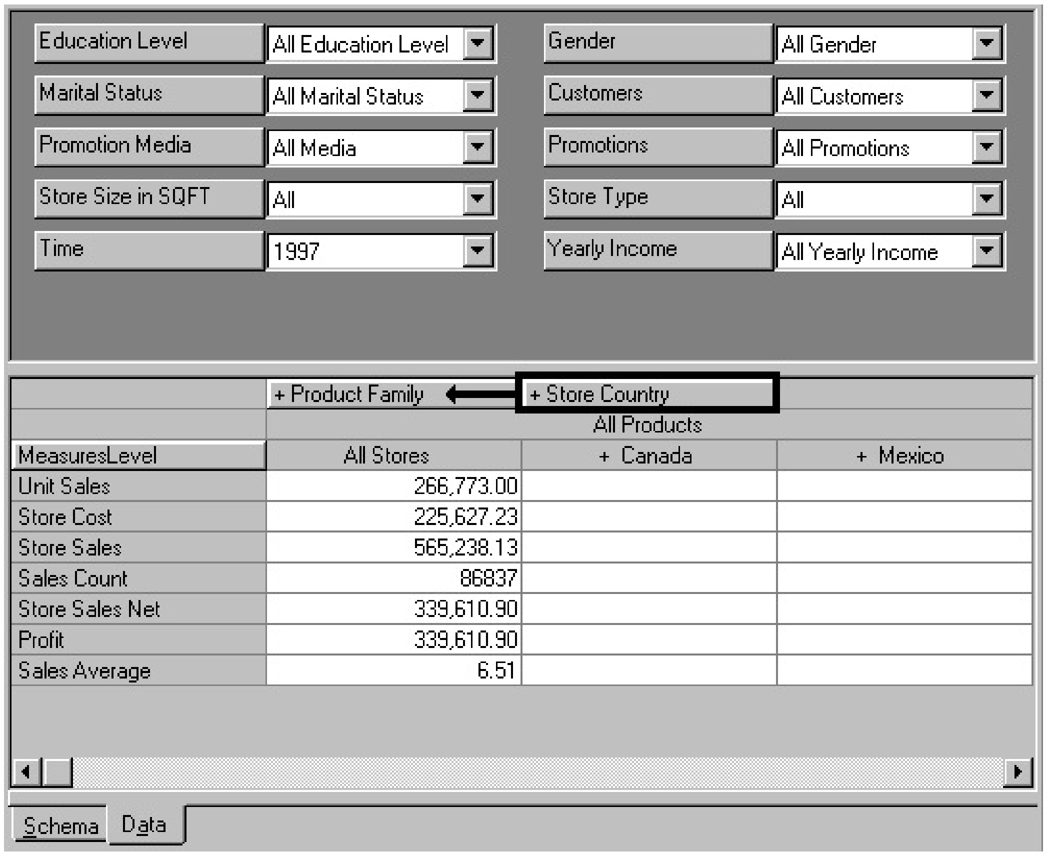
-or-

Move a slicing dimension to a column.

The new dimension appears under (or beside) the existing dimension, expanding your data viewing pane.

To change the order of dimensions in the data viewing pane

- Drag the dimension and drop it on the dimension where you want it to be located.



Education Level	All Education Level	Gender	All Gender
Marital Status	All Marital Status	Customers	All Customers
Promotion Media	All Media	Promotions	All Promotions
Store Size in SQFT	All	Store Type	All
Time	1997	Yearly Income	All Yearly Income

	+ Product Family	+ Store Country	
		All Products	
MeasuresLevel	All Stores	+ Canada	+ Mexico
Unit Sales	266,773.00		
Store Cost	225,627.23		
Store Sales	565,238.13		
Sales Count	86837		
Store Sales Net	339,610.90		
Profit	339,610.90		
Sales Average	6.51		

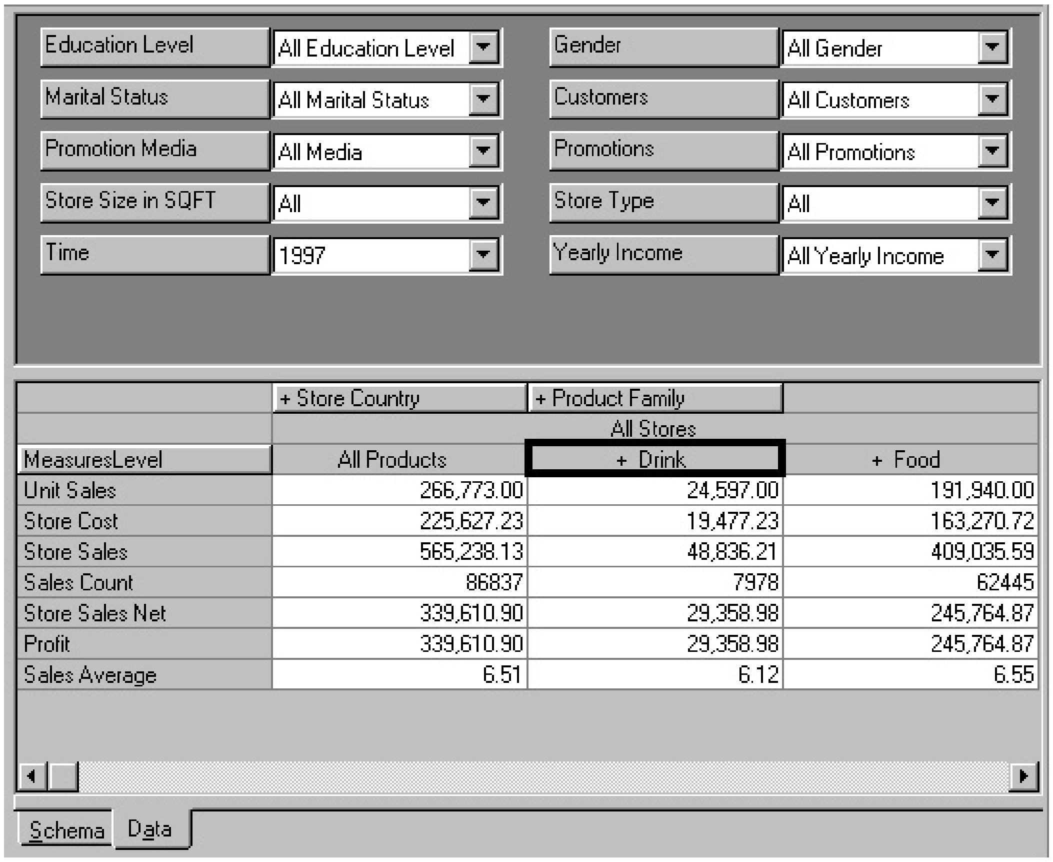
To remove a dimension from the data viewing pane

- Drag the dimension you want to remove from the data viewing pane to the data slicing pane.

To drill down into a member

- Double-click a dimension member in the data viewing pane. If the member has members beneath it, they are displayed.

  Only the member you double-click is expanded. The other members retain their current levels in the data viewing pane. In the following example, double-click the Drink member to drill down into its child members.



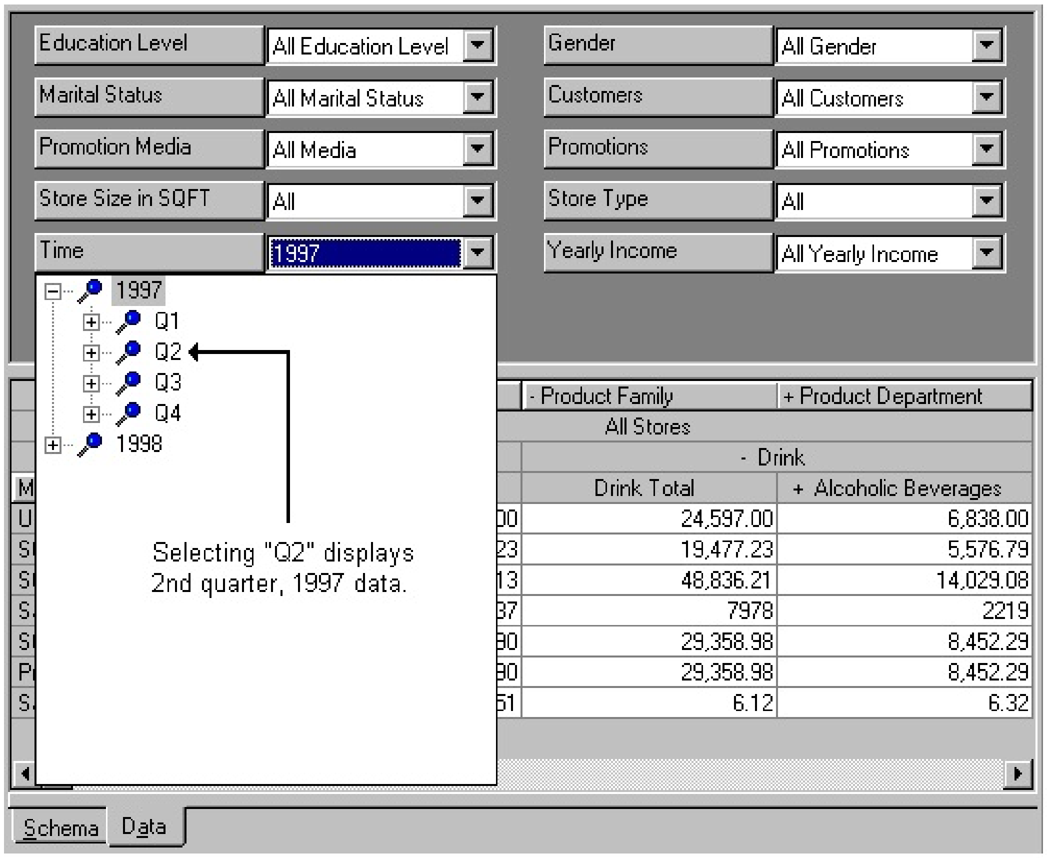
Education Level	All Education Level
Marital Status	All Marital Status
Promotion Media	All Media
Store Size in SQFT	All
Time	1997
Gender	All Gender
Customers	All Customers
Promotions	All Promotions
Store Type	All
Yearly Income	All Yearly Income

	+ Store Country	+ Product Family	
	All Stores		
MeasuresLevel	All Products	+ Drink	+ Food
Unit Sales	266,773.00	24,597.00	191,940.00
Store Cost	225,627.23	19,477.23	163,270.72
Store Sales	565,238.13	48,836.21	409,035.59
Sales Count	86837	7978	62445
Store Sales Net	339,610.90	29,358.98	245,764.87
Profit	339,610.90	29,358.98	245,764.87
Sales Average	6.51	6.12	6.55

If you want to display all the members in a level, double-click the parent level name in the data viewing pane. Level names are depicted with the shading of a button (for example, Product Family in the preceding example).

To slice through a dimension

- To change the member in a dimension in the data slicing pane, click the down arrow on the members box, expand the members, and then select a new member. The cube data in the data viewing pane changes to reflect the change in the dimension member.


Education Level	All Education Level
Marital Status	All Marital Status
Promotion Media	All Media
Store Size in SQFT	All
Time	1997
Gender	All Gender
Customers	All Customers
Promotions	All Promotions
Store Type	All
Yearly Income	All Yearly Income

		- Product Family	+ Product Department
		All Stores	
		- Drink	
M		Drink Total	+ Alcoholic Beverages
U	00	24,597.00	6,838.00
S	23	19,477.23	5,576.79
S	13	48,836.21	14,029.08
S.	37	7978	2219
S	30	29,358.98	8,452.29
P	30	29,358.98	8,452.29
S.	51	6.12	6.32

The **Data** tab has an internal memory limit that you may reach if you attempt to browse too much data or drill down too deeply. When you reach the limit, the following message is displayed:

Unable to display current view of cube.
Unable to Allocate Memory For Flexgrid.

You cannot increase the limit by adding or allocating more memory. If you reach the limit, reduce the amount or depth of data you are attempting to browse or use another browser.

Write Enable Dialog Box

Use this dialog box to enable read/write access to a cube.

If you write-enable a cube, end users in roles granted read/write access to the cube's cells can record changes to cell data while they browse the cube. Changes are recorded in a table called a writeback table, separate from the cube data and its source data. End users who browse a write-enabled cube see the net effect of all changes in the writeback table for the cube.

This dialog box appears when, in the Analysis Manager tree pane, you right-click a cube and then click **Write-Enable**.

Options

Table name

Type a name for the cube's writeback table.

Data source

Select the data source to store the writeback table.

New

Click to specify a new data source.

Analysis Services

OLE DB Compliance

This section contains an overview of the differences between the features of this version of Microsoft® SQL Server™ 2000 Analysis Services and the OLAP section of the OLE DB specification of March 1999 (version 2.6).

Analysis Services complies with all but 4 of the 394 mandatory items required by the specification.

Analysis Services implements all but one of the seven optional items identified in the Multidimensional Expressions (MDX) grammar Backus-Naur form (BNF) specification.

Analysis Services extends the MDX grammar with 32 functions and aliases.

Analysis Services implements the optional **IMDFind** interface as defined in the specification.

Noncompliant Items

The following items are part of the OLE DB specification that Analysis Services either implements differently from the specification, or omits entirely.

Mandatory

Of the four mandatory items that do not comply with the OLE DB specification in Analysis Services, one is an MDX function, one is a member function of a COM interface, and the last two are related to MDX grammar. The following table identifies each item and explains the difference between the specification and Analysis Services implementation.

Item	Analysis Services implementation	OLE DB specification
Ancestors(«Member», «Level»)	«Level» must be a level of the same hierarchy as «Member».	The set of returned members must all be from the same

		hierarchy, but «Level» need not be a level of the same hierarchy as «Member».
Count(«Set»[, EXCLUDEEMPTY | INCLUDEEMPTY])	Empty cells are counted by default.	Empty cells are not counted by default.
Descendants(«Member», «Level»[, «Desc_flags»]	Returns leaf members on all levels between member and specified level.	Returns leaf members irrespective of the level.
Descendants(«Member», «Distance»[, «Desc_flags»]	Returns leaf members on all levels between member and specified distance in hierarchy.	Returns leaf members irrespective of the distance.
IMDRangeRowset::GetRangeRowset	Returns S_OK only if the requested range returns a single cell or the entire dataset. Any other range returns E_FAIL.	Partial datasets are allowed.
Expression delimiters	Single quotation marks are required around expressions that define calculated members in MDX statements.	No delimit characters are required.
Cell Property Value delimiters	Single quotation marks are required around cell property values in MDX statements.	No delimit characters are required.

Optional

Of the seven optional MDX functions identified in the BNF specification, Analysis Services implements all but one. This omitted function is the CASE statement, both simple and searched forms.

Additional Items and Extensions

Analysis Services provides additional functionality that extends the OLE DB specification by implementing an optional interface and providing 32 additional MDX functions and aliases.

IMDFind

IMDFind is an optional interface on the dataset object. This interface contains methods that find the ordinal number of a cell in a dataset and the ordinal number of a tuple on an axis.

Method	Description
FindCell	Finds and returns a cell ordinal based on an array of members and a starting cell ordinal in the dataset.
FindTuple	Returns a pointer to a tuple ordinal based on an array of members, a starting ordinal, and an axis identifier.

MDX Extensions

These MDX functions are available for use with Analysis Services even though they do not appear in the OLE DB specification. Most provide additional functionality, and some act as aliases for existing functions.

Function	Description
AddCalculatedMembers(«Set»)	Adds calculated members to a set.
«Member».Children	Returns the children of a member.
CovarianceN(«Set», «Numeric	Returns the covariance of two series

Expression»[, «Numeric Expression»])	evaluated over a set (unbiased).
Crossjoin(«Set1», «Set2»)	Returns the cross product of two sets.
«Set».Current	Returns the current tuple from a set during an iteration.
Dimensions(«Numeric Expression»)	Returns the dimension whose zero-based position within the cube is specified by a numeric expression.
Dimensions(«String Expression»)	Returns the dimension whose name is specified by a string.
DrilldownLevel(«Set», , «Index»)	Drills down into a specified dimension in the set.
Head(«Set»[, « Numeric Expression »])	Returns the first specified number of elements in a set.
Levels(«String Expression»)	Returns the level whose name is specified by a string expression.
«Hierarchy».Levels(«Numeric Expression»)	Returns the level whose position in a hierarchy is specified by a numeric expression.
Members(«String Expression»)	Returns the member whose name is specified by a string expression.
«Level».Ordinal	Returns the zero-based ordinal value associated with a level.
SetToArray(«Set»[, «Set»]...[, «Numeric Expression»])	Converts one or more sets to an array for use in a user-defined function.
SetToStr(«Set»)	Constructs a string from a set.
Stddev(«Set»[, «Numeric Expression»])	Alias for **Stdev**.
StddevP(«Set»[, «Numeric Expression»])	Alias for **StdevP**.
StdevP(«Set»[, «Numeric Expression»])	Returns the standard deviation of a numeric expression evaluated over a set (biased).
StripCalculatedMembers(«Set»)	Removes calculated members from a set.

StrToSet(«String Expression»)	Constructs a set from a string expression.
StrToTuple(«String Expression»)	Constructs a tuple from a string.
Subset («Set», «Start»[, «Count»])	Returns a subset of elements from a set.
Tail(«Set»[, «Count»])	Returns a subset from the end of a set.
TupleToStr(«Tuple»)	Constructs a string from a tuple.
«Dimension».UniqueName	Returns the unique name of a dimension.
«Level».UniqueName	Returns the unique name of a level.
«Member».UniqueName	Returns the unique name of a member.
ValidMeasure(«Tuple»)	Returns a valid measure in a virtual cube by forcing inapplicable dimensions to their top level.
Variance(«Set»[, «Numeric Expression»])	Alias for **Var**.
VarianceP(«Set»[, «Numeric Expression»])	Alias for **VarP**.
VarP(«Set»[, «Numeric Expression»])	Returns the variance of a numeric expression evaluated over a set (biased).
VisualTotals(«Set», «Pattern»)	Dynamically totals child members specified in a set using a pattern for the total label in the result set.

Analysis Services

OLE DB for Data Mining Compliance

This section contains an overview of the differences between the features of this version of Microsoft® SQL Server™ 2000 Analysis Services and the OLE DB for Data Mining specification of June 2000.

Analysis Services complies with all but eight items required by the specification.

Analysis Services extends the MINING_MODELS schema rowset with an additional column.

Noncompliant Items

Of the eight items that do not comply with the OLE DB for Data Mining specification, four deal directly with supported data mining grammar, two refer to supported data mining model settings, and two refer to data mining functions.

Item	Analysis Services implementation	OLE DB for Dat specification
Refining Mining Models	All of the data mining model content must be deleted and the data mining model must be retrained using the full set of old and new cases.	A data mining mo executing anothe statement with ad mining algorithm refining data min
CONTENT_TYPE	CONTENT_TYPE supports the following content types: • CONTINUOUS • DISCRETE • DISCRETIZED	CONTENT_TYP following conten • CONTIN • CYCLIC • DISCRE • DISCRE

	- KEY - TABLE	- KEY - ORDER - ORDER - PROBAI - PROBAI - PROBAI - SEQUEI - SEQUEI - STDEV - SUPPOI - TABLE - VARIAN
DISTRIBUTION_FLAG	DISTRIBUTION_FLAG supports the following distribution types: - NORMAL	DISTRIBUTION following distribu - BINOMI

		• LOG_N • MULTI • NORM • POISSO • T-DIST • UNIFO
INSERT Statement	<source data query> supports the following source data query types: • OPENROWSET • SHAPE • SINGLETON SELECT	<source data que following source • OPENR • SELEC • SHAPE • SINGLE • SINGLE
SELECT Statement	SELECT * FROM <model> syntax not allowed.	SELECT * FRO used to browse a
CREATE MINING MODEL Statement	The following modeling flags are supported in the column definition statement for handling missing values:	The following m supported in the statement for ha NOT NULL

	NOT NULL	IGNORE NULL NULL INFORM
PredictScore(<scalar column reference>**)**	Not implemented.	For predictable c returns the predic value, of the inpu specified column
PredictNodeID(<scalar column reference>**)**	Not implemented.	For predictable c returns the node I where the case is specified column
PredictProbabilityStdev(<scalar column reference>**)**	Not implemented.	This function retu deviation of prob histogram entry v probability.
PredictProbabilityVariance(<scalar column reference>**)**	Not implemented.	This function retu probability for th with the highest p

Additional Items and Extensions

Analysis Services provides additional functionality that extends the OLE DB for Data Mining specification by implementing an additional schema rowset column.

Schema Rowset Extensions

The MINING_MODELS schema rowset stores meta data related to data mining models. The following table contains columns added to the MINING_MODELS schema rowset.

Column name	Type indicator	Description
MSOLAP_MODEL_SOURCE	DBTYPE_WSTR	Stores the name of the source cube for OLAP mining models.

Analysis Services

MDX

The Multidimensional Expressions (MDX) language is used to manipulate multidimensional information in Microsoft® SQL Server™ 2000 Analysis Services. MDX is defined in the OLAP extensions in OLE DB.

Similar to SQL in many respects, MDX provides a rich and powerful syntax for the retrieval and manipulation of multidimensional data, such as the data stored in cubes on the Analysis server. Analysis Services supports MDX functions in the definitions of calculated members, as well as a full language implementation for building local cubes and querying cube data using PivotTable® Service with OLE DB and Microsoft ActiveX® Data Objects (ADO).

Additionally, MDX supports the creation and registration of user-defined functions. You can create user-defined functions to operate on multidimensional data and accept arguments and return values in the MDX syntax.

The following topics provide more information about MDX.

Topic	Description
MDX Overview	Describes basic MDX concepts and provides a comparison between SQL syntax and MDX syntax.
Basic MDX	Gives a basic overview of the construction of a simple MDX query.
Advanced MDX	Details more advanced information, such as named sets and calculated members, for complex MDX queries.
Effective MDX	Provides a list of tips, workarounds, and feature discussions regarding MDX.
MDX Functions in Analysis Services	Details the statements and functions supported by MDX.

Analysis Services

MDX Overview

This section introduces Multidimensional Expressions (MDX) and explains some of the concepts behind its structure and syntax. It contains the following topics.

Topic	Description
Introduction to MDX	Provides a brief introduction to MDX.
Key Concepts in MDX	Explains the differences between relational and multidimensional queries.
Comparison of SQL and MDX	Gives a more detailed description of the differences between SQL and MDX.

Analysis Services

Introduction to MDX

MDX, an acronym for **M**ulti**d**imensional E**x**pressions, is a syntax that supports the definition and manipulation of multidimensional objects and data. MDX is similar in many ways to the Structured Query Language (SQL) syntax, but is not an extension of the SQL language; in fact, some of the functionality that is supplied by MDX can be supplied, although not as efficiently or intuitively, by SQL.

As with an SQL query, each MDX query requires a data request (the SELECT clause), a starting point (the FROM clause), and a filter (the WHERE clause). These and other keywords provide the tools used to extract specific portions of data from a cube for analysis. MDX also supplies a robust set of functions for the manipulation of retrieved data, as well as the ability to extend MDX with user-defined functions.

MDX, like SQL, provides data definition language (DDL) syntax for managing data structures. There are MDX commands for creating (and deleting) cubes, dimensions, measures, and their subordinate objects.

Analysis Services

Key Concepts in MDX

The purpose of Multidimensional Expressions (MDX) is to make accessing data from multiple dimensions easier and more intuitive.

Dimensions, Levels, Members, and Measures

Most languages used for data definition and manipulation, such as SQL, are designed to retrieve data in two dimensions: a column dimension and a row dimension. The following diagram illustrates a traditional relational database, used to store order information.

Each table represents two-dimensional data. At the intersection of each row and column is a single element of data, called a field. The specific columns to be viewed in an SQL query are specified with a SELECT statement, and the rows to be retrieved are limited by a WHERE clause.

Multidimensional data, on the other hand, can be represented by structures with more than two dimensions. These structures, called cubes, have multiple dimensions. At the intersection of dimensions in a cube, there may be more than one element of data, called a measure. The following diagram illustrates a cube that employs three dimensions, Route, Service and Time; and two measures, Packages and Last. Each dimension is broken down into different levels, each of which is broken down further into members. For example, the Source dimension supplies the Eastern Hemisphere level, which is broken down into four members, Africa, Asia, Australia, and Europe.

As you can see, the querying of even simple data out of a multidimensional data source can be a complex task. A cube can have more than three dimensions, for example, or it may only have one dimension.

The concepts of cubes, dimensions, levels, members, and measures are important to the understanding of MDX syntax. Further reading on these architectural topics is recommended if you are new to online analytical processing (OLAP) databases.

Cells, Tuples, and Sets

As SQL returns a subset of two-dimensional data from tables, MDX returns a subset of multidimensional data from cubes.

The cube diagram illustrates that the intersection of multidimensional members creates cells from which you can obtain data. To identify and extract such data, whether it be a single cell or a block of cells, MDX uses a reference system called tuples. Tuples list dimensions and members to identify individual cells as well as larger sections of cells in the cube; because each cell is an intersection of all the dimensions of the cube, tuples can uniquely identify every cell in the cube. For the purposes of reference, measures in a cube are treated as a private dimension, named Measures, in the cube itself. For example, in the preceding diagram, the following tuple identifies a cell in which the value is 400:

```
(Source.[Eastern Hemisphere].Africa, Time.[2nd half].[4th quarter], R
```

The tuple uniquely identifies a section in the cube; it does not have to refer to a specific cell, nor does it have to encompass all of the dimensions in a cube. The following examples are all tuples of the cube diagram:

```
(Source.[Eastern Hemisphere])
(Time.[2nd half], Source.[Western Hemisphere])
```

These tuples provide sections of the cube, called slices, that encompass more than one cell.

An ordered collection of tuples is referred to as a set. In an MDX query, axis and slicer dimensions are composed of such sets of tuples. The following example is a description of a set of tuples in the cube in the diagram:

```
{ (Time.[1st half].[1st quarter]), Time.[2nd half].[3rd quarter]) }
```

In addition, it is possible to create a named set. A named set is a set with an alias, used to make your MDX query easier to understand and, if it is particularly complex, easier to process.

Axis and Slicer Dimensions

In SQL, it is usually necessary to restrict the amount of data returned from a query on a table. For example, you may want to see only two fields of a table with forty fields, and you want to see them only if a third field meets a specific criteria. You can accomplish this by specifying columns in the SELECT statement, using a WHERE statement to restrict the rows that are returned based on specific criteria.

In MDX, those concepts also apply. A SELECT statement is used to select the dimensions and members to be returned, referred to as axis dimensions. The WHERE statement is used to restrict the returned data to specific dimension and member criteria, referred to as a slicer dimension. An axis dimension is expected to return data for multiple members, while a slicer dimension is expected to return data for a single member.

The terms "axis dimension" and "slicer dimension" are used to differentiate the dimensions of the cells in the source cube of the query, indicated in the FROM clause, from the dimensions of the cells in the result cube, which can be composed of multiple cube dimensions.

Calculated Members

Calculated members are members that are based not on data, but on evaluated expressions in MDX. They are returned in the same fashion as a normal member. MDX supplies a robust set of functions that can be used to create calculated members, giving you extensive flexibility in the manipulation of multidimensional data.

User-Defined Functions

MDX provides extensibility in the form of user-defined functions using any programming language that can support Component Object Model (COM) interfaces. You create and register your own functions that operate on multidimensional data as well as accept arguments and return values in the MDX syntax. You can call user-defined functions from within Calculated Member Builder, data definition language (DDL) statements that support MDX, and MDX queries.

PivotTable Service

In Microsoft® SQL Server™ 2000 Analysis Services, MDX data definition and manipulation services are provided through PivotTable® Service. PivotTable Service also provides stand-alone OLE DB provider capabilities for multidimensional queries when not connected to an Analysis server. PivotTable Service is used for the definition and manipulation of local cubes, which can be used to locally store data in a multidimensional format.

See Also

Axis and Slicer Dimensions

Calculated Members

Cubes

Dimensions

Levels

Measures

Members, Tuples, and Sets

PivotTable Service

Analysis Services

Comparison of SQL and MDX

The Multidimensional Expressions (MDX) syntax appears, at first glance, to be remarkably similar to the syntax of Structured Query Language (SQL). In many ways, the functionality supplied by MDX is also similar to that of SQL; with effort, you can even duplicate some of the functionality provided by MDX in SQL.

However, there are some striking differences between SQL and MDX, and you should be aware of these differences at a conceptual level. The following information is intended to provide a guide to these conceptual differences between SQL and MDX, from the point of view of an SQL developer.

The principal difference between SQL and MDX is the ability of MDX to reference multiple dimensions. Although it is possible to use SQL exclusively to query cubes in Microsoft® SQL Server™ 2000 Analysis Services, MDX provides commands that are designed specifically to retrieve data as multidimensional data structures with almost any number of dimensions.

SQL refers to only two dimensions, columns and rows, when processing queries. Because SQL was designed to handle only two-dimensional tabular data, the terms "column" and "row" have meaning in SQL syntax.

MDX, in comparison, can process one, two, three, or more dimensions in queries. Because multiple dimensions can be used in MDX, each dimension is referred to as an axis. The terms "column" and "row" in MDX are simply used as aliases for the first two axis dimensions in an MDX query; there are other dimensions that are also aliased, but the alias itself holds no real meaning to MDX. MDX supports such aliases for display purposes; many OLAP tools are incapable of displaying a result set with more than two dimensions.

In SQL, the SELECT clause is used to define the column layout for a query, while the WHERE clause is used to define the row layout. However, in MDX the SELECT clause can be used to define several axis dimensions, while the WHERE clause is used to restrict multidimensional data to a specific dimension or member.

In SQL, the WHERE clause is used to filter the data returned by a query. In MDX, the WHERE clause is used to provide a slice of the data returned by a query. While the two concepts are similar, they are not equivalent.

The SQL query uses the WHERE clause to contain an arbitrary list of items that should (or should not) be returned in the result set. While a long list of conditions in the filter can narrow the scope of the data that is retrieved, there is no requirement that the elements in the clause will produce a clear and concise subset of data.

In MDX, however, the concept of a slice means that each member in the WHERE clause identifies a distinct portion of data from a different dimension. Because of the organizational structure of multidimensional data, it is not possible to request a slice for multiple members of the same dimension. Because of this, the WHERE clause in MDX can provide a clear and concise subset of data.

The process of creating an SQL query is also different than that of creating an MDX query. The creator of an SQL query visualizes and defines the structure of a two-dimensional rowset and writes a query on one or more tables to populate it. In contrast, the creator of an MDX query usually visualizes and defines the structure of a multidimensional dataset and writes a query on a single cube to populate it. This could result in a multidimensional dataset with any number of dimensions; a one-dimensional dataset is possible, for example.

The visualization of an SQL result set is intuitive; the set is a two-dimensional grid of columns and rows. The visualization of an MDX result set is not as intuitive, however. Because a multidimensional result set can have more than three dimensions, it can be challenging to visualize the structure. To refer to such two-dimensional data in SQL, the name of a column and the unique identification of a row, in whatever method is appropriate for the data, are used to refer to a single cell of data, called a field. However, MDX uses a very specific and uniform syntax to refer to cells of data, whether the data forms a single cell or a group of cells.

Although SQL and MDX share similar syntax, the MDX syntax is remarkably robust, and it can be complex. However, because MDX was designed to provide a simple, effective way of querying multidimensional data, it addresses the conceptual differences between two-dimensional and multidimensional querying in a consistent and easily understood fashion.

Analysis Services

Basic MDX

Multidimensional Expressions (MDX) commands allow you to query multidimensional objects, such as cubes, and return multidimensional datasets. This topic and its subtopics provide an overview of MDX queries.

As is the case with SQL, the author of an MDX query must determine the structure of the requested dataset before writing the query. The following topics describe MDX queries and the datasets they produce, and provide more detailed information about basic MDX syntax.

Topic	Description
The Basic MDX Query	Provides basic syntax information for an MDX query.
Members, Tuples, and Sets	Gives a brief description of members, tuples, and sets, including conceptual information and syntax.
Axis and Slicer Dimensions	Describes the axis and slicer dimensions and their use within an MDX query with the SELECT and WHERE clauses.
Establishing Cube Context	Provides a description of the purpose of the FROM clause in MDX queries.

Analysis Services

The Basic MDX Query

A basic Multidimensional Expressions (MDX) query is structured in a fashion similar to the following example:

```
SELECT [<axis_specification>
       [, <axis_specification>...]]
  FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
```

Basic MDX Syntax - SELECT Statement

In MDX, the SELECT statement is used to specify a dataset containing a subset of multidimensional data. To discuss the various syntax elements of the MDX SELECT statement, this topic presents a basic MDX query example and breaks it down into its syntax elements, discussing the purpose and structure of each element.

To specify a dataset, an MDX query must contain information about:

- The number of axes. You can specify up to 128 axes in an MDX query.
- The members from each dimension to include on each axis of the MDX query.
- The name of the cube that sets the context of the MDX query.
- The members from a slicer dimension on which data is sliced for members from the axis dimensions.

This information can be complex. As you will see in this topic, MDX syntax can provide such information in a simple and straightforward manner, using the MDX SELECT statement.

Basic MDX Query Example

The following MDX query example is used to discuss the various parts of basic MDX SELECT statement syntax:

```
SELECT
   { [Measures].[Unit Sales], [Measures].[Store Sales] } ON COLUMN
   { [Time].[1997], [Time].[1998] } ON ROWS
FROM Sales
WHERE ( [Store].[USA].[CA] )
```

The basic MDX SELECT statement contains a SELECT clause and a FROM clause, with an optional WHERE clause.

The SELECT clause determines the axis dimensions of an MDX SELECT statement. Two axis dimensions are defined in the MDX query example. For more information about the construction of axis dimensions in a SELECT clause, see Specifying the Contents of an Axis Dimension.

The FROM clause determines which multidimensional data source is to be used when extracting data to populate the result set of the MDX SELECT statement. For more information about the FROM clause, see SELECT Statement.

The WHERE clause optionally determines which dimension or member to use as a slicer dimension; this restricts the extracting of data to a specific dimension or member. The MDX query example uses a WHERE clause to restrict the data extract for the axis dimensions to a specific member of the Store dimension. For more information about the construction of a slicer dimension in a WHERE clause, see Specifying the Contents of a Slicer Dimension.

The MDX SELECT statement supports other optional syntax, such as the WITH keyword, and the use of MDX functions to construct members by calculation for inclusion in an axis or slicer dimension. For more information about the MDX SELECT statement, see SELECT Statement.

The syntax format of the MDX SELECT statement is similar to that of SQL syntax; however, you will note several obvious differences:

- MDX syntax distinguishes sets by surrounding tuples or members with braces (the { and } characters.) For more information about member, tuple, and set syntax, see Members, Tuples, and Sets.

- MDX queries can have up to 128 axis dimensions in the SELECT statement, but only the first 5 axes have aliases. An axis can be referred to by its ordinal position within an MDX query or by its alias, if it has an alias assigned to it. In the MDX query example, the COLUMNS and ROWS axis aliases are used. The MDX query could also have been written in the following fashion, using the ordinal position of each axis:

  ```
  SELECT
     { [Measures].[Unit Sales], [Measures].[Store Sales] } ON AX
     { [Time].[1997], [Time].[1998] } ON AXIS(1)
  FROM Sales
  WHERE ( [Store].[USA].[CA] )
  ```

- As with an SQL query, the FROM clause names the source of the data for the MDX query. However, unlike an SQL query, the FROM clause in an MDX query is restricted to a single cube. Information from other cubes can be retrieved, however, on a value-by-value basis using the **LookupCube** function.

- The WHERE clause is used to describe the slicer dimensions. If a dimension is not mentioned as part of the WHERE clause, Microsoft® SQL Server™ 2000 Analysis Services assumes that any dimension not assigned to an axis dimension is a slicer dimension, and the dimension is filtered on its default members. The WHERE clause can change the filtering process for specified dimensions, allowing fine control of included data.

Analysis Services

Members, Tuples, and Sets

Before proceeding on the creation of a Multidimensional Expressions (MDX) query, you should understand the definitions of members, tuples and sets, as well as the MDX syntax used to construct and refer to these elements.

Members

A member is an item in a dimension representing one or more occurrences of data. Think of a member in a dimension as one or more records in the underlying database whose value in this column falls under this category. A member is the lowest level of reference when describing cell data in a cube.

For example, the following diagram is shaded to represent the Time.[2nd half].[3rd quarter] member.

The bracket characters, [and], are used if the name of a member has a space or a number in it. Although the Time dimension is one word, bracket characters can also be used around it as well; the member shown in the previous diagram could also be represented as:

```
[Time].[2nd half].[4th quarter]
```

Member Names and Member Keys

A member can be referenced by either its member name or by its member key. The previous example referenced the member by its member name, 4th quarter, in the Time dimension. However, the member name can be duplicated in the case of dimensions with nonunique member names, or it can be changed in the case of changing dimensions.

An alternate method to reference members is by referencing the member key. The member key is used by the dimension to specifically identify a given member. The ampersand (**&**) character is used in MDX to differentiate a member key from a member name, as shown in the following example:

```
[Time].[2nd half].&[Q4]
```

In this case, the member key of the 4th quarter member, Q4, is used. Referencing the member key ensures proper member identification in changing dimensions and in dimensions with nonunique member names.

The ampersand character can be used to indicate a member key reference in any MDX expression.

Calculated Members

Members can also be created, as part of an MDX query, to return data based on evaluated expressions instead of stored data in a cube to be queried. These members are called calculated members, and they provide a great deal of the power and flexibility of MDX. The WITH keyword is used in an MDX query to define a calculated member. For example, if you want to provide a forecast estimate all of the packages by adding 10% of the existing value of the Packages measure, you can simply create a calculated member that provides the information and use it just like any other member in the cube, as demonstrated in the following example.

```
WITH MEMBER [Measures].[PackagesForecast] AS
'[Measures].[Packages] * 1.1'
```

For more information, see Calculated Members.

Member Functions

MDX supplies a number of functions for retrieving members from other MDX entities, such as dimensions and levels, so that explicit references to a member are not always necessary. For example, the **FirstChild** function allows the retrieval of all the members from a given dimension or level; to get the first child member of the Time dimension, you can explicitly state it, as demonstrated in the following example:

```
Time.[1st half]
```

You can also use the **FirstChild** function to return the same member, demonstrated in the next example.

```
Time.FirstChild
```

For more information about MDX member functions, see MDX Function List.

Tuples

A tuple is used to define a slice of data from a cube; it is composed of an ordered collection of one member from one or more dimensions. A tuple is used to identify specific sections of multidimensional data from a cube; a tuple composed of one member from each dimension in a cube completely describes a cell value. Put another way, a tuple is a vector of members; think of a tuple as one or more records in the underlying database whose value in these columns falls under these categories. A series of diagrams presents different types of tuples.

The shaded area of the cube represents the (Time.[2nd half]) tuple. Note that this tuple encompasses half of the cube, because it does not rule out any information in the Source or Route dimensions.

The following diagram is shaded to represent the (Time.[2nd half], Route.nonground.air) tuple.

This tuple represents the cells at the intersection of these members.

In MDX, tuples are syntactically constructed depending upon their complexity. If a tuple is composed of only one member from a single dimension, often referred to as a simple tuple, the following syntax is acceptable.

```
Time.[2nd half]
```

If a tuple is composed of members from more than one dimension, the members represented by the tuple must be enclosed in parentheses, as demonstrated in the following example.

```
(Time.[2nd half], Route.nonground.air)
```

A tuple composed of a single member can also be enclosed in parentheses, but this is not required. Tuples are often grouped together in sets for use in MDX queries.

Tuple Functions

There are a few MDX functions that return tuples, and they can be used anywhere that a tuple is accepted.

For more information about tuple functions, see MDX Function List.

Tuples and Dimensionality

A tuple can encompass members in multiple dimensions, as well as multiple members from the same dimension. The term *dimensionality* is used to indicate the dimensions described by the members in a tuple. Order plays a factor in the dimensionality of a tuple, and can affect the use of a tuple within a set.

Sets

A set is an ordered collection of zero, one or more tuples. A set is most commonly used to define axis and slicer dimensions in an MDX query, and as such may have only a single tuple or may be, in certain cases, empty. The following example shows a set of two tuples:

```
{ (Time.[1st half], Route.nonground.air), (Time.[2nd half], Route.nong
```

A set can contain more than one occurrence of the same tuple. The following set is acceptable:

```
{ Time.[2nd half], Time.[2nd half] }
```

A set refers to either a set of member combinations, represented as tuples, or to the values in the cells that the tuples in the set represent, depending on the context of usage for the set.

In MDX syntax, tuples are enclosed in braces to construct a set.

IMPORTANT Sets composed of a single tuple are not tuples; they are interpreted as sets by MDX. Certain MDX functions accept tuples as parameters, and will raise an error if a single tuple set is passed. Tuples and single-tuple sets are not interchangeable.

Set Functions

Explicitly typing tuples and enclosing them in braces is not the only way to retrieve a set. MDX supports a wide variety of functions that return sets.

The colon operator allows you to use the natural order of members to create a set. For example, the following set:

```
{[1st quarter]:[4th quarter]}
```

retrieves the same set of members as the following set:

```
{[1st quarter], [2nd quarter], [3rd quarter], [4th quarter]}
```

The colon operator is an inclusive function; the members on both sides of the colon operator are included in the resulting set.

Other MDX functions that return sets can be used either by themselves or as part of a comma-delimited list of members. For example, all of the following MDX expressions are valid:

```
{Time.Children}
{Time.Children, Route.nonground.air}
{Time.Children, Route.nonground.air, Source.Children}
```

For more information about set functions, see MDX Function List.

Sets and Dimensionality

Like tuples, sets also have dimensionality. As a set is composed of tuples, so the dimensionality of a set is expressed by the dimensionality of each tuple within it. Because of this, tuples within a set must have the same dimensionality. In other words, this example would not work as a set:

```
{ (Time.[2nd half], Route.nonground.air), (Route.nonground.air, Time.
```

The order of tuples in a set is important; it affects, for example, the nesting order in an axis dimension. The first tuple represents the first, or outermost, dimension, the second tuple represents the next outermost dimension, and so on.

Named Sets

A named set is a set for which an alias has been created. A named set is most commonly used in complex MDX queries to make these queries easier to read and to increase the ease of maintenance.

For more information about named sets, see Building Named Sets in MDX.

Analysis Services

Axis and Slicer Dimensions

When formulating a Multidimensional Expressions (MDX) query, an application typically looks at the cubes and divides the set of dimensions into two subsets:

- Axis dimensions, for which data is retrieved for multiple members.
- Slicer dimensions, for which data is retrieved for a single member.

Because axis and slicer dimensions can be constructed from multiple dimensions of the cube to be queried, these terms are used to differentiate the dimensions employed by the cube to be queried from the dimensions created in the cube returned by an MDX query.

For example, assume that a cube exists, named TestCube, with two simple dimensions named Route and Time. Because the measures of the cube are part of the Measures dimension, this cube has three dimensions in all. The query is to provide a matrix in which the Packages measure can be compared across routes and times.

In the following MDX query example, the Route and Time dimensions are used as axis dimensions and the Measures dimension is used as the slicer dimension. The Members function indicates that the members of the dimension or level are to be used to construct a set, instead of having to explicitly state each member of a given dimension or level in an MDX query.

```
SELECT
  { Route.nonground.Members } ON COLUMNS,
  { Time.[1st half].Members } ON ROWS
FROM TestCube
WHERE ( [Measures].[Packages] )
```

The resulting grid of values would resemble the following table, showing the value of the Packages measure at each intersection of the COLUMNS and ROWS axis dimensions.

	air	**sea**

1st quarter	60	50
2nd quarter	45	45

MDX evaluates the axis and slicer dimensions first, building the structure of the result cube before retrieving the information from the cube to be queried.

The slicer dimension is similar to an axis dimension in its purpose, but has limitations that axis dimensions do not share.

Note Microsoft® SQL Server™ 2000 Analysis Services supports a maximum of 128 shared or private dimensions in a cube, in addition to the Measures dimension. Therefore, MDX queries on Analysis Services cubes are limited to 129 axes maximum.

See Also

Specifying the Contents of an Axis Dimension

Specifying the Contents of a Slicer Dimension

Specifying the Contents of an Axis Dimension

Axis dimensions determine the edges of a multidimensional result set. Multidimensional Expressions (MDX) uses the SELECT clause to specify axis dimensions by assigning a set to a particular axis. The following information describes how this assignment is handled in MDX.

In the following syntax example, each <axis_specification> value defines one axis dimension. The number of axes in the dataset is equal to the number of <axis_specification> values in the Multidimensional Expressions (MDX) query. An MDX query can support up to 128 specified axes, but very few MDX queries will use more than 5 axes.

The breakdown of the <axis_specification> syntax is:

```
<axis_specification> ::= <set> ON <axis_name>

<axis_name> ::= COLUMNS | ROWS | PAGES | SECTIONS | CHAP
```

Each axis dimension is associated with a number: 0 for the x-axis, 1 for the y-axis, 2 for the z-axis, and so on. The <index> value is the axis number. For the first 5 axes, the aliases COLUMNS, ROWS, PAGES, SECTIONS, and CHAPTERS can be used in place of AXIS(0), AXIS(1), AXIS(2), AXIS(3), and AXIS(4), respectively.

An MDX query cannot skip axes. That is, a query that includes one or more <axis_name> values must not exclude lower-numbered or intermediate axes. For example, a query cannot have a ROWS axis without a COLUMNS axis, or have COLUMNS and PAGES axes without a ROWS axis.

However, you can specify a SELECT clause with no axes (that is, an empty SELECT clause). In this case, all dimensions are slicer dimensions, and the MDX query selects one cell.

Each <set> value defines the contents of the axis. For more information about sets, see Members, Tuples, and Sets.

Specifying the Contents of a Slicer Dimension

Slicer dimensions filter multidimensional data. You can use them to limit the data returned by including them in the WHERE clause of a Multidimensional Expressions (MDX) query.

Dimensions that are not explicitly assigned to an axis are assumed to be slicer dimensions and filter with their default members. The default member of a dimension can be explicitly specified in its **Default Member** property in Analysis Manager. This property is equivalent to the **DefaultMember** property in Decision Support Objects (DSO). If no default member is explicitly specified, the default member is the All member if an (All) level exists, or else an arbitrary member of the highest level. (The name of the All member is not necessarily All.)

Slicer dimensions can also be specified explicitly by using the WHERE clause of the MDX syntax. The breakdown of the WHERE clause syntax is:

```
[WHERE [<slicer_specification>]]
```

The member name [All] will probably not be unique within the cube, because many dimensions possess an [All] level. It is recommended that you qualify it with the dimension name to make it unambiguous. The following example demonstrates the use of the WHERE clause and the All member:

```
WHERE ( [Route].[All], [Time].[1st half] )
```

A slicer dimension can accept only expressions that evaluate into a single tuple. This does not mean that only a single tuple can be explicitly stated in the slicer dimension, as the following example shows:

```
WHERE ( [Time].[1st half], [Route].[nonground] )
```

If a set of tuples is supplied as the slicer expression, MDX will attempt to evaluate the set, aggregating the result cells in every tuple along the set. In other words, MDX will attempt to use the **Aggregate** function on the set, aggregating each measure by its associated aggregation function. The following examples show a valid WHERE clause using a set of tuples:

```
WHERE { ([Time].[1st half], [Route].[nonground]), ([Time].[1st half],
```

If the «slicer_specification» cannot be resolved into a single tuple, an error will occur.

For more information about the **Aggregate** function, see Aggregate.

Analysis Services

Establishing Cube Context

To establish cube context, indicate the cube on which you want the Multidimensional Expressions (MDX) query to run. The FROM clause in an MDX query determines the cube context. The following syntax indicates which cube supplies the context for the MDX query:

```
FROM «cube_specification»
```

The «cube_specification» is completed with the name of a single cube.

For example, if an MDX query is to be run against the SalesCube cube, the FROM clause would be:

```
FROM SalesCube
```

This does not limit you from working with more than one cube at a time; you can use the **LookupCube** function to retrieve data from cubes outside the cube context. The following syntax will cause an error, because unlike SQL, the FROM clause in an MDX query does not usually permit joins:

```
FROM SalesCube, OtherCube
```

However, some OLAP providers may permit the joining of cubes along congruent dimensions; if two cubes share a dimension, the cubes can be joined using this syntax, in a fashion similar to that of linked cubes. For more information about joining cubes, see your OLAP provider documentation.

For more information about the FROM clause in the MDX SELECT statement, see SELECT Statement.

Analysis Services

Advanced MDX

The Multidimensional Expressions (MDX) syntax is designed not only to extract simple data from multidimensional data sources, but also to provide additional functionality to create named sets, calculated members, and write information back to dimensions and cells.

The following topics cover the more advanced aspects of MDX syntax.

Topic	Description
Creating and Using Property Values	Details the process of creating and using dimension, level, member, and cell properties
Building Named Sets in MDX	Describes the purpose of named sets in MDX and the techniques needed to create and use them in MDX queries
Building Calculated Members in MDX	Provides information about calculated members in MDX, including the techniques needed to create and use them in MDX expressions
Building Caches in MDX	Gives information about caches in MDX, including the techniques needed to create and use them in MDX queries
Building Calculated Cells in MDX	Details the process of creating and using calculated cells
Creating and Using User-Defined Functions in MDX	Details the creation and use of external function libraries and user-defined functions in MDX
Using Writebacks	Describes the technique of writing information back into the dimensions and cells of multidimensional data sources using MDX
Using DRILLTHROUGH to Retrieve Source Data	Discusses the use of the MDX DRILLTHROUGH statement to retrieve the rowsets of source data applicable to a

	cell in a multidimensional data source
Understanding Pass Order and Solve Order	Details the concepts of pass order and solve order, and how these features affect MDX queries and expressions

Analysis Services

Creating and Using Property Values

Multidimensional Expressions (MDX) supports intrinsic and custom properties for dimensions, levels, members, and cells. The intrinsic properties are used to provide unique names, captions, and even formatting and font sizes for individual cells. Custom properties, on the other hand, can be used to provide almost any kind of additional attribute to members.

The following table lists the topics that describe the use of such properties.

Topic	Description
Using Member Properties	Describes the intrinsic properties supported by members, the process of creating new member properties, and the process used to query them
Using Cell Properties	Describes the intrinsic properties supported by cells, the process of creating new cell properties, and the process used to query them

Using Member Properties

In the axis specification for a given axis, the set expression selects tuples to populate the axis. The dataset returns some basic information about each member in each tuple, such as the member name, parent level, the number of children, and so on. These are referred to as member properties. Members often have additional properties associated with them, and member properties are available for all members at a given level. In terms of organization, member properties are treated as dimensionally organized data, stored on a single dimension.

For example, the Products level may offer the SKU, SRP, Weight, and Volume properties for each product. These properties are not members, but contain additional information about members at the Products level. All members support intrinsic member properties, such as the formatted value of a member, while dimensions and levels supply additional intrinsic dimension and level member properties, such as the ID of a member. Additional member properties can be created in Analysis Manager using Dimension Editor or Cube Editor, or with Multidimensional Expressions (MDX) statements. Member properties can be retrieved through the use of the DIMENSION PROPERTIES keyword or the **Properties** function.

DIMENSION PROPERTIES Keyword

An application might want to extend member information by adding member properties on the axis. Therefore, each level of each dimension may contain a set of available properties for the members.

The DIMENSION PROPERTIES keyword is used to specify member properties to be used for a given axis dimension. The following syntax defines the MDX SELECT syntax, adding the syntax for the DIMENSION PROPERTIES keyword:

```
SELECT [<axis_specification>
       [, <axis_specification>...]]
  FROM [<cube_specification>]
```

```
[WHERE [<slicer_specification>]]
```

The <axis_specification> value includes an optional <dim_props> value, which enables querying of dimension, level, and member properties using the DIMENSION PROPERTIES keyword. The breakdown of the <axis_specification> syntax with the <dim_props> value is:

```
<axis_specification> ::= <set> [<dim_props>] ON <axis_name>
```

The <set> and <axis_name> values are described in Specifying the Contents of an Axis Dimension. The breakdown of the <dim_props> syntax is:

```
<dim_props> ::= [DIMENSION] PROPERTIES <property> [,<propert
```

The breakdown of the <property> syntax varies depending on the property you are querying. Intrinsic member properties for dimensions and levels must be preceded with the name of the dimension and/or level. Intrinsic member properties for members cannot be qualified by the dimension or level name. Custom member properties should be preceded by the name of the level in which they reside.

Additional member properties can be selected by using the DIMENSION PROPERTIES keyword after the set expression of the axis specification. For example, the following MDX query:

```
SELECT
   CROSSJOIN(Years, (Sales, BudgetedSales)) ON COLUMNS,
   NON EMPTY Products.MEMBERS
   DIMENSION PROPERTIES Products.SKU, Products.SRP ON ROV
FROM SalesCube
WHERE (January, SalesRep.[All], Geography.USA)
```

returns the following dataset:

You can specify only those dimension properties projected on the axis for that particular axis. You can mix requests for intrinsic dimension and level member properties in the same query with intrinsic member properties. The difference between intrinsic dimension and level member properties and intrinsic member properties is explained in greater detail later in this topic.

Properties Function

Member properties can also be retrieved by the use of the **Properties** function in MDX. For example, the following MDX query uses the WITH keyword to create a calculated member consisting of the [Store Sqft] member property:

```
WITH
   MEMBER [Measures].[Store Size] AS
   'Val(Store.CurrentMember.Properties("Store Sqft"))'

SELECT
   {[Measures].[Unit Sales], [Measures].[Store Size]} ON COLUMNS,
   {[Store].[Store Name].Members} ON ROWS
From Sales
```

to generate a result set similar to the one in the following table:

For more information about building calculated members, see Building Calculated Members in MDX.

Note the use of the **Val**() function in the MDX query example. The **Properties** function is a string function; all member properties retrieved with the **Properties** function will be coerced into strings.

See Also

CREATE CUBE Statement

Custom Member Properties

Intrinsic Dimension and Level Member Properties

Intrinsic Member Properties

Properties

SELECT Statement

Intrinsic Dimension and Level Member Properties

All dimensions and levels support a list of intrinsic member properties, displayed in the following table. These member properties are used in the context of a specific dimension or level, and supply values for each member of the specified dimension or level. For example, specifying the following statement in a Multidimensional Expressions (MDX) query:

```
[Sales].Name
```

returns the names of each referenced member of the [Sales] dimension.

Property	Description
ID	The internally maintained ID for the member
Key	The value stored in the MEMBER_KEY column of the MEMBERS schema rowset for the member
Name	The name of the member

Dimension member properties are preceded by the name of the dimension to which the property applies. The following example demonstrates the appropriate syntax:

```
DIMENSION PROPERTIES «Dimension».ID
```

Level member properties can be preceded with the level name or, for additional specification, the dimension and level name, as shown here:

```
DIMENSION PROPERTIES [«Dimension».]«Level».ID
```

See Also

Using Member Properties

Intrinsic Member Properties

All members support a list of intrinsic member properties as well, displayed in the following table. Intrinsic member properties cannot be requested for a specific dimension or level; they apply to all members of an axis dimension in a Multidimensional Expressions (MDX) query. Specifying, for example, the following statement in an MDX query:

```
PROPERTIES DESCRIPTION
```

returns the description of each member in the axis dimension.

The following table lists the intrinsic member properties supported by Microsoft® SQL Server™ 2000 Analysis Services.

Property	Description
CALCULATION_PASS_DEPTH	For calculated cells only. The pass depth fo the calculation formula, this property determines how many passes are needed to resolve the calculation formula. For more information about pass order, see Understanding Pass Order and Solve Orde
CALCULATION_PASS_NUMBER	For calculated cells only. The pass number for the calculation formula, this property determines on which pass the calculation formula will begin evaluation and end calculation. The default for this property is its maximum value is 65,535. For more information about pass order, see Understanding Pass Order and Solve Orde
CATALOG_NAME	The name of the catalog to which this member belongs.
CHILDREN_CARDINALITY	The number of children that the member h This can be an estimate, so you should not rely on this to be the exact count. Provider should return the best estimate possible

	should return the best estimate possible.
CONDITION	For calculated cells only. The calculation condition of the calculated cells. This property receives an MDX logical expression, which is evaluated on each cel in the calculation subcube. If it returns Tru the calculation formula is applied and the c returns the resulting value. If it returns Fals the cell returns the original cell value. If no specified, CONDITION defaults to True (i other words, the calculation formula applie to all cells in the calculation subcube.)
CUBE_NAME	The name of the cube to which this membe belongs.
DESCRIPTION	A human-readable description of the meml or calculated cells definition.
DIMENSION_UNIQUE_NAME	The unique name of the dimension to whic this member belongs. For providers that generate unique names by qualification, ea component of this name is delimited.
DISABLED	For calculated cells only. A Boolean prope that indicates whether or not the calculated cells are disabled. DISABLED defaults to False.
HIERARCHY_UNIQUE_NAME	The unique name of the hierarchy. If the member belongs to more than one hierarch there is one row for each hierarchy to whic it belongs. For providers that generate uniq names by qualification, each component of this name is delimited.
LEVEL_NUMBER	The distance of the member from the root the hierarchy. The root level is zero.
LEVEL_UNIQUE_NAME	Unique name of the level to which the member belongs. For providers that genera unique names by qualification, each component of this name is delimited.
MEMBER_CAPTION	A label or caption associated with the

	member. It is used primarily for display purposes. If a caption does not exist, MEMBER_NAME is returned.
MEMBER_GUID	The member GUID.
MEMBER_NAME	The name of the member.
MEMBER_ORDINAL	The ordinal number of the member. This is the sort rank of the member when member of this dimension are sorted in their natura sort order. If providers do not have the concept of natural ordering, this should be the rank when sorted by MEMBER_NAM
MEMBER_TYPE	The type of the member. It can be one of th following values: • MDMEMBER_TYPE_REGULA • MDMEMBER_TYPE_ALL • MDMEMBER_TYPE_FORMUL • MDMEMBER_TYPE_MEASUR • MDMEMBER_TYPE_UNKNOV MDMEMBER_TYPE_FORMULA takes precedence over MDMEMBER_TYPE_MEASURE. Therefore, if there is a formula (calculated member on the Measures dimension, it is listed as MDMEMBER_TYPE_FORMUL
MEMBER_UNIQUE_NAME	The unique name of the member. For providers that generate unique names by qualification, each component of this name delimited.

PARENT_COUNT	The number of parents that this member ha
PARENT_LEVEL	The distance of the member's parent from t root level of the hierarchy. The root level i zero.
PARENT_UNIQUE_NAME	The unique name of the member's parent. NULL is returned for any members at the root level. For providers that generate uniq names by qualification, each component of this name is delimited.
SCHEMA_NAME	The name of the schema to which this member belongs.

Columns in the MEMBERS schema rowset support the intrinsic member properties. For more information about the MEMBERS schema rowset, see MDSCHEMA_MEMBERS. Other intrinsic member properties can be supported, depending upon the provider. However, all providers must support the intrinsic member properties listed here to be compliant with the OLAP section of the OLE DB specification dated March 1999 (2.6).

Intrinsic member properties are used without additional specification of any sort, as intrinsic member properties apply to all members. The following syntax example demonstrates usage:

PROPERTIES «Property»

IMPORTANT Because intrinsic member properties cannot be qualified by the dimension or level name, a consumer cannot choose different intrinsic member properties for different dimensions (or levels) on an axis. For example, if the ROWS axis has Geography and SalesRep dimensions, the consumer cannot choose the MEMBER_CAPTION intrinsic member property for the Geography dimension or the MEMBER_UNIQUE_NAME intrinsic member property for the SalesRep dimension. The consumer must choose the same intrinsic member property (or properties) for all dimensions on an axis.

See Also

Using Member Properties

Custom Member Properties

Custom member properties can be added to a specific named level in a dimension. Custom member properties cannot be added to the (All) level of a dimension, or to the dimension itself. Custom member properties can be added to server based dimensions or cubes using Dimension Editor or Cube Editor in Analysis Manager, or by an application using the Decision Support Objects (DSO) library. Additionally, custom member properties can be defined as part of the CREATE CUBE statement when creating local cubes in PivotTable® Service.

The syntax used to refer to custom member properties is similar to that used to refer to intrinsic level member properties, as demonstrated in the following example:

```
PROPERTIES [«Dimension».]«Level».«Custom Member Property»
```

See Also

Using Member Properties

Using Cell Properties

Cell properties in Multidimensional Expressions (MDX) contain information about the content and format of cells in a multidimensional data source, such as a cube. MDX supports the CELL PROPERTIES keyword in an MDX SELECT statement to retrieve intrinsic cell properties. Intrinsic cell properties are most commonly used to assist in the visual presentation of cell data.

The following example displays the syntax of the MDX SELECT statement, with the CELL PROPERTIES keyword syntax included.

```
SELECT [<axis_specification>
       [, <axis_specification>...]]
  FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
[<cell_props>]
```

The syntax of the <cell_props> value is displayed here, and it employs the CELL PROPERTIES keyword along with one or more intrinsic cell properties:

```
<cell_props> ::= CELL PROPERTIES <property> [, <property>...]
```

The supported intrinsic cell properties used in the <property> value are listed in the following table, with brief descriptions on the content of the cell property.

Property	Description
BACK_COLOR	The background color for displaying the **VALUE** or **FORMATTED_VALUE** property. For more information, see FORE_COLOR and BACK_COLOR Contents.
CELL_EVALUATION_LIST	The semicolon-delimited list of evaluated formulas applicable to the cell, in order from lowest to highest solve order. For more information about solve order, see

	Understanding Pass Order and Solve Order
CELL_ORDINAL	The ordinal number of the cell in the dataset.
FORE_COLOR	The foreground color for displaying the **VALUE** or **FORMATTED_VALUE** property. For more information, see FORE_COLOR and BACK_COLOR Contents.
FONT_NAME	The font to be used to display the **VALUE** or **FORMATTED_VALUE** property.
FONT_SIZE	Font size to be used to display the **VALUE** or **FORMATTED_VALUE** property.
FONT_FLAGS	The bitmask detailing effects on the font. The value is the result of a bitwise OR operation of one or more of the following constants: • MDFF_BOLD = 1 • MDFF_ITALIC = 2 • MDFF_UNDERLINE = 4 • MDFF_STRIKEOUT = 8 For example, the value 5 represents the combination of bold (MDFF_BOLD) and underline (MDFF_UNDERLINE) font effects.
FORMAT_STRING	The format string used to create the **FORMATTED_VALUE** property value. For more information, see FORMAT_STRING Contents.

FORMATTED_VALUE	The character string that represents a formatted display of the **VALUE** property.
NON_EMPTY_BEHAVIOR	The measure used to determine the behavior of calculated members when resolving empty cells.
SOLVE_ORDER	The solve order of the cell.
VALUE	The unformatted value of the cell.

Providers are not required to support all intrinsic cell properties; only the **CELL_ORDINAL**, **FORMATTED_VALUE**, and **VALUE** cell properties must be supported. All cell properties, intrinsic or provider-specific, are defined in the PROPERTIES schema rowset, including their data types and provider support. For more information about the PROPERTIES schema rowset, see MDSCHEMA_PROPERTIES.

By default, if the CELL PROPERTIES keyword is not used, the cell properties returned are **VALUE**, **FORMATTED_VALUE**, and **CELL_ORDINAL** (in that order). If the CELL PROPERTIES keyword is used, only those cell properties explicitly stated with the keyword are returned.

The following example demonstrates the use of the CELL PROPERTIES keyword in an MDX query:

```
SELECT
   {[Measures].[Unit Sales], [Measures].[Store Size]} ON COLUMNS,
   {[Store].[Store Name].Members} ON ROWS
FROM Sales
CELL PROPERTIES VALUE, FORMATTED_VALUE, FORMAT_ST
```

Cell properties are not returned for MDX queries that return flattened rowsets; in this case, each cell is represented as if only the **FORMATTED_VALUE** cell property were returned.

Custom Member Options

Cell properties can be set through Analysis Manager by using the **Custom Member Options** property of Dimension Editor or Cube Editor. The **Custom**

Member Options property accepts a column reference containing, for each member, a comma-delimited list of cell properties. The cell properties are represented as string expressions, shown in the following example.

```
FORE_COLOR='255',BACK_COLOR='65535'
```

The example will provide, for the specified member, a yellow background with a red foreground. Cell properties usually roll up to parent members, unless the parent is a custom member with cell properties. In this case, the parent cell properties override the cell properties derived from its children.

See Also

Properties Pane (Dimension Editor Schema View)

Properties Pane (Cube Editor Schema View)

FORMAT_STRING Contents

The cell property **FORMAT_STRING** is used to format the **VALUE** cell property, creating the value for the **FORMATTED_VALUE** cell property. The **FORMAT_STRING** cell property handles both string and numeric raw values, applying a format expression against the value to return a formatted value for the **FORMATTED_VALUE** cell property. The following tables detail the syntax and formatting characters used to handle string and numeric values.

String Values

A format expression for strings can have one section or two sections separated by a semicolon (;).

Usage	Result
One section	The format applies to all string values.
Two sections	The first section applies to string data, whereas the second section applies to null values and zero-length strings ("").

The characters described in the following table can appear in the format string for character strings.

Character	Description
@	Character placeholder. It displays a character or a space. If the string has a character in the position where the at sign (**@**) appears in the format string, it displays the character. Otherwise, it displays a space in that position. Placeholders are filled from right to left unless there is an exclamation point (**!**) in the format string.
&	Character placeholder. It displays a character or nothing. If the string has a character in the position where the ampersand (**&**) appears, it displays the character. Otherwise, it displays nothing. Placeholders are filled from right to left unless there is an exclamation point (**!**) in the format string.
<	Forces lowercase. It displays all characters in lowercase

	format.
>	Forces uppercase. It displays all characters in uppercase format.
!	Forces left-to-right fill of placeholders. (The default is to fill placeholders from right to left.)

Numeric Values

A user-defined format expression for numbers can have anywhere from one to four sections separated by semicolons. If the format argument contains one of the named numeric formats, only one section is allowed.

Usage	**Result**
One section	The format expression applies to all values.
Two sections	The first section applies to positive values and zeros, the second to negative values.
Three sections	The first section applies to positive values, the second to negative values, and the third to zeros.
Four sections	The first section applies to positive values, the second to negative values, the third to zeros, and the fourth to null values.

The following example has two sections: The first section defines the format for positive values and zeros, and the second section defines the format for negative values.

"$#,##0;($#,##0)"

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value. For example, the following format displays positive and negative values using the format in the first section and displays "Zero" if the value is zero:

"$#,##0;;\Z\e\r\o"

The following table identifies the characters that can appear in the format string for number formats.

Character	**Description**
None	Displays the number with no formatting.
0	Digit placeholder. Displays a digit or a zero. If the expression has a digit in the position where the 0 appears in the format string, it displays the digit. Otherwise, it displays a zero in that position. If the number has fewer digits than there are zeros (on either side of the decimal) in the format expression, it displays leading or trailing zeros. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, it rounds the number to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, it displays the extra digits without modification.
#	Digit placeholder. Displays a digit or nothing. If the expression has a digit in the position where the # appears in the format string, it displays the digit. Otherwise, it displays nothing in that position. This symbol works like the 0 digit placeholder except that leading and trailing zeros are not displayed if the number has the same or fewer digits than there are # characters on either side of the decimal separator in the format expression.
.	Decimal placeholder. (In some locales, a comma is used as the decimal separator.) The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only number signs (#) to the left of this symbol, numbers smaller than 1 begin with a decimal separator. To display a leading zero displayed with fractional numbers, use 0 as the first

	digit placeholder to the left of the decimal separator. The actual character used as a decimal placeholder in the formatted output depends on the number format recognized by your system.
%	Percentage placeholder. The expression is multiplied by 100. The percent character (**%**) is inserted in the position where it appears in the format string.
,	Thousand separator. (In some locales, a period is used as a thousand separator.) The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a thousand separator surrounded by digit placeholders (**0** or **#**). Two adjacent thousand separators, or a thousand separator immediately to the left of the decimal separator (whether or not a decimal is specified), means "scale the number by dividing it by 1000, rounding as needed." For example, you can use the format string "**##0**,," to represent 100 million as 100. Numbers smaller than 1 million are displayed as 0. Two adjacent thousand separators in any position other than immediately to the left of the decimal separator are treated simply as specifying the use of a thousand separator. The actual character used as the thousand separator in the formatted output depends on the number format recognized by your system.
:	Time separator. (In some locales, other characters may be used to represent the time separator.) The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings.
/	Date separator. (In some locales, other characters may be used to represent the date separator.) The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings.
E- E+ e- e+	Scientific format. If the format expression contains at least

	one digit placeholder (**0** or **#**) to the right of **E-**, **E+**, **e-**, or **e+**, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use **E-** or **e-** to place a minus sign next to negative exponents. Use **E+** or **e+** to place a minus sign next to negative exponents and a plus sign next to positive exponents.
- + $ ( )	Displays a literal character. To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (**" "**).
\	Displays the next character in the format string. To display a character that has special meaning as a literal character, precede it with a backslash (\). The backslash itself is not displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\). Examples of characters that cannot be displayed as literal characters are the date-formatting and time-formatting characters (**a**, **c**, **d**, **h**, **m**, **n**, **p**, **q**, **s**, **t**, **w**, **y**, /, and **:**), the numeric-formatting characters (**#**, **0**, **%**, **E**, **e**, **comma**, and **period**), and the string-formatting characters (**@**, **&**, <, >, and **!**).
"ABC"	Displays the string inside the double quotation marks (**" "**). To include a string in format from within code, use Chr(**34**) to enclose the text. (The character code for a double quotation mark is **34**.)

Date Values

The following table identifies characters that can appear in the format string for date/time formats.

Character	**Description**
:	Time separator. (In some locales, other characters may be used to represent the time separator.) The time separator

	separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings.
/	Date separator. (In some locales, other characters may be used to represent the date separator.) The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings.
C	Displays the date as **ddddd** and displays the time as **ttttt**, in that order. Displays only date information if there is no fractional part to the date serial number. Displays only time information if there is no integer portion.
d	Displays the day as a number without a leading zero (1–31).
dd	Displays the day as a number with a leading zero (01–31).
ddd	Displays the day as an abbreviation (Sun–Sat).
dddd	Displays the day as a full name (Sunday–Saturday).
ddddd	Displays the date as a complete date (including day, month, and year), formatted according to your system's short date format setting. For Microsoft® Windows®, the default short date format is **m/d/yy**.
dddddd	Displays a date serial number as a complete date (including day, month, and year), formatted according to the long date setting recognized by your system. For Windows, the default long date format is **mmmm dd, yyyy**.
w	Displays the day of the week as a number (1 for Sunday through 7 for Saturday).
ww	Displays the week of the year as a number (1–54).
m	Displays the month as a number without a leading zero (1–12). If **m** immediately follows **h** or **hh**, the minute rather than the month is displayed.
mm	Displays the month as a number with a leading zero (01–12). If m immediately follows **h** or **hh**, the minute rather than the month is displayed.
mmm	Displays the month as an abbreviation (Jan–Dec).
mmmm	Displays the month as a full month name (January–

	December).
q	Displays the quarter of the year as a number (1–4).
y	Displays the day of the year as a number (1–366).
yy	Displays the year as a two-digit number (00–99).
yyyy	Displays the year as a four-digit number (100–9999).
h	Displays the hour as a number without leading zeros (0–23).
hh	Displays the hour as a number with leading zeros (00–23).
n	Displays the minute as a number without leading zeros (0–59).
nn	Displays the minute as a number with leading zeros (00–59).
s	Displays the second as a number without leading zeros (0–59).
ss	Displays the second as a number with leading zeros (00–59).
t t t t t	Displays a time as a complete time (including hour, minute, and second), formatted using the time separator defined by the time format recognized by your system. A leading zero is displayed if the leading zero option is selected and the time is earlier than 10:00 (for example 09:59), in either the A.M. or the P.M. cycle. For Windows, the default time format is **h:mm:ss**.
AM/PM	Uses the 12-hour clock. Displays an uppercase with any hour from midnight until noon; displays an uppercase **AMPM** with any hour from noon until midnight.
am/pm	Uses the 12-hour clock. Displays a lowercase **am** with any hour from midnight until noon; displays a lowercase **pm** with any hour from noon until midnight.
A/P	Uses the 12-hour clock. Displays an uppercase **A** with any hour from midnight until noon; displays an uppercase **P** with any hour from noon until midnight.
a/p	Uses the 12-hour clock. Displays a lowercase **a** with any hour from midnight until noon; displays a lowercase **p** with any hour from noon until midnight.
AMPM	Uses the 12-hour clock. Displays the AM string literal as

	defined by your system with any hour from midnight until noon; displays the PM string literal as defined by your system with any hour from noon until midnight. **AMPM** can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. For Windows, the default format is **AM/PM**.

FORE_COLOR and BACK_COLOR Contents

The **FORE_COLOR** and **BACK_COLOR** cell properties are used to store color information for the text and the background of a cell, respectively, in the Microsoft® Windows® operating system red-green-blue (RGB) format.

The valid range for a normal RGB color is 0 to 16,777,215 (&H00FFFFFF). The high byte of a number in this range always equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

For example, the value 255 (&H000000FF) represents red, the value (65280 (&H0000FF00) represents green, and the value 16711680 (&H00FF0000) represents blue.

Analysis Services

Building Named Sets in MDX

A set in Multidimensional Expressions (MDX) can be a lengthy and complex declaration, and difficult to follow or understand. For example, the following MDX query examines the unit sales of the various Chardonnay and Chablis wines in **FoodMart 2000**:

```
SELECT
   {[Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   {Measures.[Unit Sales]} ON ROWS
FROM Sales
```

The MDX query, although fairly simple in terms of the result set, is lengthy and unwieldy when it comes to maintenance.

One method of easing maintenance and increasing understandability of an MDX query such as the previous example is to create a named set. A named set is simply a set expression associated with an alias. A named set can incorporate member or function that can normally be incorporated into a set. The named set alias is treated as a set expression, and can be used anywhere a set expression is accepted.

To illustrate, the previous MDX query example is rewritten to employ a named set, as shown in the following example:

```
WITH SET [ChardonnayChablis] AS
   '{[Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
```

```
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
  [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W

SELECT
  [ChardonnayChablis] ON COLUMNS,
  {Measures.[Unit Sales]} ON ROWS
FROM Sales
```

The WITH keyword is used to create the [ChardonnayChablis] named set, which is then reused in the MDX SELECT statement. In this fashion, the set created with the WITH keyword can be changed without disturbing the MDX SELECT statement. For more information about using the WITH keyword to create named sets, see Using WITH to Create Named Sets.

The named set makes the MDX query example a bit easier to follow, but still difficult to maintain because the named set is defined as part of the MDX query itself. The scope of the named set is limited to this MDX query alone, and is not reusable.

MDX and PivotTable® Service, however, offer the capability of creating a named set with a wider scope. The CREATE SET statement allows the client application to create a named set that exists for the lifetime of the MDX session, making the named set available to all MDX queries in that session. The CREATE SET statement makes sense, for example, in a client application that consistently reuses a set in a variety of queries. For more information about using the CREATE SET to create named sets in a session, see CREATE SET Statement.

Even this scope, however, may be limiting in terms of maintenance. Microsoft® SQL Server™ 2000 Analysis Services offers the capability of creating global named sets, stored as part of a cube. For more information about creating global named sets, see Creating Named Sets.

See Also

PivotTable Service

Using WITH to Create Named Sets

The WITH keyword is included as part of the MDX SELECT statement, to allow construction of named sets as part of an MDX query.

The following syntax is used to add the WITH keyword to the MDX SELECT statement:

```
[WITH <formula_specification>]
      [, <formula_specification>]
SELECT [<axis_specification>
       [, <axis_specification>...]]
  FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
```

The <formula_specification> value for named sets is further broken out in the following syntax definition:

```
<formula_specification> ::= SET <set_name> AS '<set>'
```

The <set_name> parameter contains the alias for the named set. The <set> parameter contains the set expression to which the named set alias will refer.

For example, the [ChardonnayChablis] named set is used to refer specifically to all of the Chardonnay and Chablis wine members in the Product dimension of the **FoodMart 2000** database. The syntax for the named set is depicted in the following example:

```
WITH SET [ChardonnayChablis] AS
   '{[Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and 
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
   [Product].[All Products].[Drink].[Alcoholic Beverages].[Beer and W
```

You can also use MDX functions in the set expression used to create a named set. The following MDX query example uses the **Filter**, **CurrentMember**, **Name**, and **InStr** functions to create the [ChardonnayChablis] named set, as used in earlier MDX query examples in this topic.

```
WITH SET [ChardonnayChablis] AS
   'Filter([Product].Members, (InStr(1, [Product].CurrentMember.Name

SELECT
   [ChardonnayChablis] ON COLUMNS,
   {Measures.[Unit Sales]} ON ROWS
FROM Sales
```

Analysis Services

Building Calculated Members in MDX

In Multidimensional Expressions (MDX), a calculated member is defined as a member that is resolved not by retrieving data, but by calculating an MDX expression to return a value. This innocuous definition covers an incredible amount of ground; the ability to construct and use calculated members in an MDX query provides a great deal of manipulation capability for multidimensional data. This topic discusses some of the simpler aspects of creating calculated members, as covered in the following table.

Topic	Description
Using WITH to Create Calculated Members	Discusses the use of the WITH keyword to create calculated members in an MDX query.
Using Functions in Calculated Members	Details the use of functions in calculated members and other MDX expressions.
Conditional Expressions	Covers the use of conditional expressions, such as the IF keyword, in calculated members.

Using WITH to Create Calculated Members

Similar to the way it is used in named sets, the WITH keyword in Multidimensional Expressions (MDX) is used to describe calculated members.

The following syntax is used to add the WITH keyword to the MDX SELECT statement:

```
[WITH <formula_specification>]
     [, <formula_specification>]
SELECT [<axis_specification>
      [, <axis_specification>...]]
  FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
```

The <formula_specification> value for calculated members is further broken out in the following syntax definition:

```
<formula_specification> ::= MEMBER <member_name>
                          AS '<value_expression>'
                          [,SOLVE_ORDER = <unsigned integer>]
                          [,<cell_property>=<value_expression>...]
```

The <member_name> value is the fully qualified name of the calculated member, including the dimension or level to which the calculated member is associated, and the <value_expression> value, after it has been evaluated, returns the value of the calculated member. Optionally, the SOLVE_ORDER keyword can be used to specify the solve order of the calculated member; if not used, the solve order of the calculated member is set by default to 0.

The values of intrinsic cell properties for a calculated member can be optionally specified by supplying the name of the cell property in the <cell_property> value and the value of the cell property in the <value_expression> value.

For example, the following MDX query example defines two calculated members. The first calculated member, [Measures].[StoreType], is used to represent the Store Type member property. The second calculated member, [Measures].[ProfitPct], is used to calculate the total profit margin for a given store, and represent it as a formatted percentile value.

```
WITH
   MEMBER [Measures].[StoreType] AS
   '[Store].CurrentMember.Properties("Store Type")',
   SOLVE_ORDER = 2
   MEMBER [Measures].[ProfitPct] AS
   'Val((Measures.[Store Sales] - Measures.[Store Cost]) / Measures.[St
   SOLVE_ORDER = 1, FORMAT_STRING = 'Percent'
SELECT
   { [Store].[Store Name].Members} ON COLUMNS,
   { [Measures].[Store Sales], [Measures].[Store Cost], [Measures].[Sto
   [Measures].[ProfitPct] } ON ROWS
FROM Sales
```

Calculated members can be created at any point within a hierarchy. For example, the following MDX query example defines a calculated member, created as a child member of the [Beer and Wine] member, to determine whether a given store has at least 100.00 in unit sales for beer and wine:

```
WITH
   MEMBER [Product].[Beer and Wine].[BigSeller] AS
  'IIf([Product].[Beer and Wine] > 100, "Yes","No")'
SELECT
   {[Product].[BigSeller]} ON COLUMNS,
   {Store.[Store Name].Members} ON ROWS
FROM Sales
```

You can also create calculated members that depend not only on existing members in a cube, but also on other calculated members defined in the same MDX expression. The following example illustrates such an MDX expression:

```
WITH
```

```
   MEMBER [Measures].[ProfitPct] AS
   'Val((Measures.[Store Sales] - Measures.[Store Cost]) / Measures.[St
   SOLVE_ORDER = 1, FORMAT_STRING = 'Percent'
   MEMBER [Measures].[ProfitValue] AS
   '[Measures].[Store Sales] * [Measures].[ProfitPct]',
   SOLVE_ORDER = 2, FORMAT_STRING = 'Currency'
SELECT
   { [Store].[Store Name].Members} ON COLUMNS,
   { [Measures].[Store Sales], [Measures].[Store Cost], [Measures].[Pro
   [Measures].[ProfitPct] } ON ROWS
FROM Sales
```

The second calculated member, [Measures].[ProfitValue], uses the value created in the first calculated member, [Measures].[ProfitPct], to generate its value.

Using Functions in Calculated Members

Calculated members in Multidimensional Expressions (MDX) are extremely flexible. One of the ways in which calculated members provide such flexibility is in the wide variety of functions available for use in MDX. Besides the intrinsic MDX functions provided by the Microsoft® SQL Server™ 2000 Analysis Services function library, calculated members can also take advantage of external function libraries to supply additional capability.

A discussion of all of the myriad ways to use calculated members is beyond the scope of this topic. Instead, this topic focuses on the most commonly employed operators and functions in calculated members, and how to use them.

Operators

MDX supports a variety of arithmetic, logical, and comparison operators for use in MDX expressions.

Arithmetic Operators

Arithmetic operators support a basic set of arithmetic operations. Arithmetic precedence is followed when resolving arithmetic operations; multiplication and division operators are processed first, followed by addition and subtraction operators. If all of the arithmetic operators used in an expression have the same order of precedence; for example, as in the statement a + b + c + d, the arithmetic operators are handled in a left to right order. The basic arithmetic operators supported are specified in the following table.

Operator	Description
+	Addition
-	Subtraction and unary negation
*	Multiplication
/	Division

Comparison Operators

Comparison operators compare two string, numeric or date expressions and return TRUE or FALSE based on the outcome of the tested comparison. For the purposes of comparison, null values are treated as zero when a null value is compared with a nonnull value. To check for null values in a cell, use the **IsEmpty** or **Is** functions to return TRUE if the cell contains a null value, FALSE otherwise. The TRUE and FALSE constants are supported; the TRUE constant evaluates to 1, while the FALSE constant evaluates to 0.

Operator	**Description**
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to
=	Equal to

Bitwise Operators

Bitwise operators return a TRUE or FALSE value based on the review of logical expressions. As the TRUE and FALSE constants are supported, either of the following MDX expressions is now valid:

```
([Measures].[IsTrue] AND [Measures].[IsFalse]) = 0
([Measures].[IsTrue] AND [Measures].[IsFalse]) = FALSE
```

Logical operators require expressions that can be evaluated to a logical value. Numeric expressions are implicitly converted to logical values before a logical comparison is performed. Any numeric expression that evaluates to 0 or NULL is considered FALSE, while any numeric expression that evaluates to something other than 0 is considered TRUE. String expressions are not implicitly converted; attempting to use a bitwise operator with string expressions will result in an error.

Operator	**Description**
«Expression1» **AND**	Returns TRUE if both expressions are

«Expression2»	true, FALSE otherwise.
«Expression1» **OR** «Expression2»	Returns TRUE if either expression is true, FALSE otherwise.
NOT «Expression1»	Returns TRUE if the expression is not true, FALSE otherwise.
«Expression1» **XOR** «Expression2»	Returns TRUE if either expression, but not both, is true, FALSE otherwise.

Set Operators

Set operators are provided to deal with the creation, separation, and joining of sets, as described in the following table.

Operator	**Description**
«Set1» **+** «Set1»	Performs the Union function on two sets.
«Set1» ***** «Set2»	Performs the Crossjoin function on two sets.
«Set1» **-** «Set2»	Performs the Except function on two sets.
«Member1»**:**«Member2»	Creates a naturally ordered set, with the two members as endpoints and all members between the two specified members included as members of the set.

Functions

MDX supplies a wide variety of functions for use in MDX expressions. This topic briefly touches on each category of functions, broken out by the type of data returned by the MDX functions in a specific category.

For more information about the categories of MDX functions, see MDX Function List.

Numeric Functions

MDX supplies a rich set of numeric functions, which can be used to perform a variety of aggregation and statistical calculations.

Aggregate functions in MDX are used to quickly perform a calculation across a number of members, usually specified as a set. For example, the **Aggregate** function aggregates the cells formed by all the members in a set, and can do so much easier than attempting to perform a manual aggregation. The **Aggregate** function is extremely powerful when combined with a measure that produces a sum, as the following MDX query example demonstrates:

```
WITH
  MEMBER [Time].[1st Half Sales] AS 'Aggregate({Time.[Q1], Time
  MEMBER [Time].[2nd Half Sales] AS 'Aggregate({Time.[Q3], Tim
  MEMBER [Time].[Difference] AS 'Time.[2nd Half Sales] - Time.[1s
SELECT
  { [Store].[Store State].Members} ON COLUMNS,
  { Time.[1st Half Sales], Time.[2nd Half Sales], Time.Difference} ON
FROM Sales
WHERE [Measures].[Store Sales]
```

The query produces the sum of the store sales for each state, with aggregations for the first and second halves of the year supplied by the first two calculated members using the **Aggregate** function, with a difference between the two supplied by a third calculated member.

MDX also supplies a list of statistical functions as well, for handling routine statistical calculations such as statistical covariance and standard deviation. For example, the **Median** function computes the median value across a set, as demonstrated in the following MDX query.

```
WITH
  MEMBER [Time].[1st Half Sales] AS 'Sum({[Time].[Q1], [Time].[C
  MEMBER [Time].[2nd Half Sales] AS 'Sum({[Time].[Q3], [Time].[
  MEMBER [Time].[Median] AS 'Median(Time.Members)'
SELECT
  NON EMPTY { [Store].[Store Name].Members} ON COLUMNS,
  { [Time].[1st Half Sales], [Time].[2nd Half Sales], [Time].[Median]}
FROM Sales
```

```
WHERE [Measures].[Store Sales]
```

In this case, the [Time].[Median] calculated member provides the median value of store sales for each store, in addition to the aggregation of store sales for each half of the year for each store provided by the [Time].[1st Half Sales] and [Time].[2nd Half Sales] calculated members.

String Functions

MDX supplies a number of string functions not just for string processing within MDX expressions, but to support user-defined functions in MDX as well. For example, the **MemberToStr** function converts a member reference to a string in the MDX format for use with a user-defined function, as user-defined functions cannot accept object references from MDX.

Set Functions

Set functions are used to return sets in MDX, giving you the capability to easily build dynamically defined sets and quickly create reusable named sets. One of the most commonly used set functions, the **Members** function, returns all members, excluding calculated members, of a level or dimension as a set. The following MDX query example shows the **Members** function in action.

```
SELECT
  NON EMPTY { [Store].[Store Name].Members} ON COLUMNS,
  {Measures.[Store Sales]} ON ROWS
FROM Sales
```

The MDX query example returns the total store sales figures for each store in the Sales cube. Without the Members function, you would have to explicitly enter each and every store name for it to function as it does in the MDX query example.

Tuple Functions

As with set functions, tuple functions are used to return tuples in MDX. Tuple functions are also supplied, such as the **StrToTuple** function, to aid user-defined functions in MDX. As user-defined functions cannot handle MDX object references, a user-defined function can pass back a string return value in MDX format, representing a tuple, and use the **StrToTuple** function to convert it to a valid tuple reference.

Member Functions

Members are often referred to in calculated members; member functions allow calculated members to perform complex member retrieval, negotiating hierarchies and sets with equal ease.

The resolution of calculated members in MDX can be iterative in nature, as calculated members can be constructed based upon iteration over the members of a set. Functions in MDX such as **CurrentMember** allow you to take advantage of this iterative capability.

Other Functions

MDX supplies other functions as well, including functions that deal with dimensions, hierarchies, levels, and arrays. For example, the **SetToArray** function allows user-defined functions to receive set references as a variant array of individual members represented as strings, allowing you to create user-defined functions that can supply set related functionality.

Conditional Expressions

Another capability in Multidimensional Expressions (MDX) is the ability to create conditional expressions, expressions that return different information depending upon a decision made in the calculated member based on the existence of a condition.

The following topics discuss the various conditional expressions in use in MDX.

Topic	Description
IIf Function	Describes the use of the **IIf** function in MDX expressions for simple decisions.

IIf Function

The **IIf** function in Multidimensional Expressions (MDX) can be used to perform simple, yes-or-no decisions. For example, consider the following MDX query example.

```
WITH MEMBER [Measures].[BigSeller] AS
   'IIf(Measures.[Store Sales] > 20000, "Yes", "No")'

SELECT
   {[Store].[Store Name].Members} ON COLUMNS,
   {[Measures].[Store Sales], [Measures].[BigSeller]} ON ROWS
FROM Sales
```

The MDX query example returns two rows for each store in the Sales cube. One row, the [Measures].[Store Sales] member, supplies the total store sales for each store. The second row is a calculated member that, based on the store sales for each store, determines if the store is a "big seller". That is, the **IIf** function is used to check a simple yes-or-no condition. In this case, the condition is whether or not the store sales figure for each store is greater than $20,000.00. If it is, the value of the member for that store is Yes. If the store sales figure is equal to or less than $20,000.00, it returns the value No.

This is a simple but graphic example of the use of the **IIf** function to return different values based upon a single Boolean condition; other MDX functions and operators can be used to supply the returned values in the **IIf** function.

For more information about the syntax of the **IIf** function, see IIf.

Analysis Services

Building Caches in MDX

Another feature Multidimensional Expressions (MDX) provides to improve performance is the ability to load a commonly used slice of a cube into memory, caching it for faster retrieval.

Microsoft® SQL Server™ 2000 Analysis Services and PivotTable® Service automatically cache query definitions, data, and meta data on the server and client sides, respectively. This caching increases performance in those cases where queries are repeatedly requesting the same data or meta data, reducing network traffic or execution time.

The ability to create caches for specific data in MDX gives you complete control over the caching of data to be used repeatedly, allowing fine-tuning of query performance.

In terms of creation scope, caches are similar to named sets in that a cache may be created for the lifetime of a single query or a session.

To create a cache to be used at the session level, the CREATE CACHE statement can be used. The CREATE CACHE statement can be used to create caches at the query level, but the WITH statement can perform this task just as easily.

For example, the following MDX query uses the WITH statement to cache:

```
WITH CACHE AS '(Store.[Store Name].Members)'

SELECT
  {[Store].[Store Name].Members} ON COLUMNS,
  {[Measures].[Unit Sales]} ON ROWS
FROM Sales
```

While the WITH statement can be used to create a cache for a single query, the CREATE CACHE statement can be used to create caches at the session level, as well. The CREATE CACHE statement requires PivotTable Service in order to employ a session level cache.

For more information about the CREATE CACHE statement, see CREATE

CACHE Statement.

Using WITH to Create Caches

As with named sets and calculated members, the WITH keyword is also used to create query level caches, usable for the lifetime of a single query. The following syntax is used to add the WITH keyword to the MDX SELECT statement:

```
[WITH <formula_specification>]
     [, <formula_specification>]
SELECT [<axis_specification>
      [, <axis_specification>...]]
  FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
```

The <formula_specification> value for caches is further broken out in the following syntax definition:

```
<formula_specification> ::= CACHE AS '(<set>[, <set>...])'
```

The <set> value is the set expression used to create the cache. The <set> value can support the use of MDX set functions.

When using the <set> set expression for constructing a cache, the following rules apply:

- Each <set> must contain members from only one dimension. Each member must be distinct.
- Each <set> must be from a different dimension.
- The <set> cannot contain measures.

Analysis Services

Building Calculated Cells in MDX

Multidimensional Expressions (MDX) provides you with a number of tools for generating calculated values, such as calculated members, custom rollups, and custom members. Although powerful and versatile features, they provide limited functionality because they affect members, not cells. It is difficult to affect a specific set of cells, or a single cell for that matter, using these features.

The calculated cells feature provides this functionality by allowing you to define a specific slice of cells, called a calculation subcube, and apply a formula to each and every cell within the calculation subcube, subject to an optional condition that can be applied to each cell.

Calculated cells take advantage of the pass order feature in Microsoft® SQL Server™ 2000 Analysis Services to provide such complex functionality as goal-seeking formulas, by allowing recursive passes to be made with calculated cells, with calculation formulas applied at specific passes in the pass order.

For more information on pass order, see Understanding Pass Order and Solve Order.

In terms of creation scope, calculated cells are similar to calculated members in that calculated cells can be made globally available as part of a cube, or temporarily created for the lifetime of either a session or a single query.

To create calculated cells as part of a cube, use the CREATE CELL CALCULATION statement. For existing cubes, the ALTER CUBE statement can also be used to add calculated cells.

To create calculated cells for the lifetime of a session, use the CREATE CELL CALCULATION statement.

To create calculated cells for the lifetime of a query, use the WITH statement.

See Also

ALTER CUBE Statement

CREATE CELL CALCULATION Statement

Using WITH to Create Calculated Cells

Using WITH to Create Calculated Cells

Similar to the way it is used in calculated members, the WITH keyword in Multidimensional Expressions (MDX) is used to describe calculated cells.

The following syntax is used to add the WITH keyword to the MDX SELECT statement:

```
[WITH <formula_specification>]
     [, <formula_specification>]
SELECT [<axis_specification>
      [, <axis_specification>...]]
  FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
```

The <formula_specification> value for calculated cells is further broken out in the following syntax definition:

```
<formula_specification> ::= CELL CALCULATION <formula_name>
                        FOR '(<calculation_subcube>)'
                        AS '<calculation_formula>'
                        [,<calculation_property_list>]
```

The <cell_property_list> is further defined by the following syntax:

```
<cell_property_list> ::= <property_name = '<value>'
                      [, <property_name> = '<value>'...]
```

The <formula_name> value is the name of the calculated cells. The <calculation_subcube> contains a list of orthogonal, single-dimensional MDX set expressions, each of which must resolve to one of the following categories of sets.

Category	Description
Empty set	An MDX set expression that resolves into an empty

	set. In this case, the set is ignored.
Single member set	An MDX set expression that resolves into a single member.
Set of level members	An MDX set expression that resolves into the members of a single level. An example of this is the «Level».**Members** MDX function. To include calculated members, use the «Level».**AllMembers** MDX function.
Set of descendants	An MDX set expression that resolves into the descendants of a specified member. An example of this is the **Descendants**(«Member», «Level», «Desc_flags») MDX function.

If a dimension is not described in the <calculation_subcube> argument, it is assumed that all members are included for the purposes of constructing the calculation subcube. Therefore, if the <calculation_subcube> argument is NULL, the calculated cells definition applies to the entire cube.

The <calculation_formula> argument contains an MDX expression that evaluates to a cell value for all of the cells defined in the <calculation_subcube> argument.

The <calculation property list> argument contains a list of member properties to be applied to the cells specified in the <calculation_subcube> argument.

The following properties apply specifically to calculated cells.

Property	Description
CALCULATION_PASS_DEPTH	The pass depth for the calculation formula, this property determines how many passes are needed to resolve the calculation formula. For more information about pass order, see Understanding Pass Order and Solve Order.
CALCULATION_PASS_NUMBER	The pass number for the calculation formula, this property determines on

	which pass the calculation formula will begin calculation. The default for this property is 1. For more information about pass order, see Understanding Pass Order and Solve Order.
CELL_EVALUATION_LIST	The semicolon-delimited list of evaluated formulas applicable to the cell, in order from lowest to highest solve order. For more information about solve order, see Understanding Pass Order and Solve Order
CONDITION	The calculation condition of the calculated cells, this property receives an MDX logical expression, which is evaluated on each cell in the calculation subcube. If it returns True, the calculation formula is applied and the cell returns the resulting value. If it returns False, the cell returns the original cell value. If not specified, CONDITION defaults to True (in other words, the calculation formula applies to all cells in the calculation subcube).
DESCRIPTION	A human-readable text description of the calculated cells definition.
DISABLED	A Boolean property which indicates whether or not the calculated cells are disabled. DISABLED defaults to False.

Other standard cell properties, such as **FORE_COLOR** and **BACK_COLOR**, can be used as well.

For more information about using cell properties and using member properties, see Using Cell Properties and Using Member Properties.

Additional Considerations

The calculation condition, specified by the **CONDITION** property, is processed only once, depending on the creation scope of the calculated cells definition. This provides increased performance for the evaluation of multiple calculated cells definitions, especially with overlapping calculated cells across cube passes.

If created at global scope, as part of a cube, the calculation condition is processed when the cube is processed. If cells are modified in the cube in any way, and the cells are included in the calculation subcube of a calculated cells definition, the calculation condition may not be accurate until the cube is reprocessed. This can occur through the use of writebacks, for example. The calculation condition is reprocessed when the cube is reprocessed.

If created at session scope, the calculation condition is processed when the statement is issued during the session. As with calculated cells definitions created globally, if the cells are modified, the calculation condition may not be accurate for the calculated cells definition.

If created at query scope, the calculation condition is processed when the query is executed. The cell modification issue applies here, as well, although data latency issues are minimal at best due to the low processing time of MDX query execution.

The calculation formula, on the other hand, is processed whenever an MDX query is issued against the cube involving cells included in the calculated cells definition, no matter the scope.

See Also

Understanding Pass Order and Solve Order

Analysis Services

Creating and Using User-Defined Functions in MDX

Multidimensional Expressions (MDX) supplies a great deal of intrinsic functions, designed to accomplish everything from standard statistical calculation to member traversal in a hierarchy. But, as with any other complex and robust product, there is always the need to extend the functionality of such a product further.

To this end, MDX provides the ability to add user-defined function references to MDX statements. This ability is already in common use in MDX; the functionality supplied by external libraries, such as the Microsoft® Excel and Microsoft Visual Basic® for Applications libraries, takes advantage of this capability.

Using a User-Defined Function in MDX

Calling a user-defined function in MDX is done in the same manner as calling an intrinsic MDX function. For a function that takes no parameters, the name of the function and an empty pair of parentheses are used, as shown here:

```
MyNewFunction()
```

If the user-defined function takes one or more parameters, then the parameters are supplied, in order, separated by commas. The following example demonstrates a sample user-defined function with three parameters:

```
MyNewFunctionWithParms("Parameter1", 2, 800)
```

USE LIBRARY Statement

Before employing a user-defined function in an MDX statement, however, the external library that contains the user-defined function must first be loaded into memory. Loading an external library is performed with the USE LIBRARY statement.

All user-defined functions must be associated with a Component Object Model (COM) class in order to be used, usually supplied in the form of a Microsoft ActiveX® dynamic link library (DLL).

If, for example, the user-defined function is part of an ActiveX DLL named MyFunc.dll, located in the C:\Winnt\System path, you can use the USE LIBRARY statement to load it by the library name, as demonstrated here:

```
USE LIBRARY "C:\WINNT\SYSTEM\MyFunc.dll"
```

The USE LIBRARY statement can also load user-defined functions by class name, as each class must be registered in order to work correctly. So, if your example function is located in the example ActiveX DLL and associated with the class "MyFuncClass", the library can be loaded using the following example:

```
USE LIBRARY "MyFunc.MyFuncClass"
```

This method is recommended when referring to libraries that may be in different locations on different servers. As ActiveX DLL components must be registered on server and client machines, referring to the class name ensures that the library is loaded from the correct location, regardless of that location.

Multiple libraries can be loaded at the same time with a single USE LIBRARY statement, by separating the library names or class names with commas, as demonstrated here:

```
USE LIBRARY "C:\WINNT\SYSTEM\MyFunc.dll", "C:\WINNT\SY
```

A USE LIBRARY statement with no parameters unregisters all function libraries except the Microsoft SQL Server™ 2000 Analysis Services function library.

PivotTable® Service supports the USE LIBRARY statement. For more information about the USE LIBRARY statement, see USE LIBRARY Statement.

DROP LIBRARY Statement

The DROP LIBRARY statement can be used to unload a specific library or to unload all libraries. As with the USE LIBRARY statement, the DROP LIBRARY syntax can accept either the file name or the class name, as demonstrated in the following statement:

```
DROP LIBRARY "MyFunc.MyFuncClass"
```

PivotTable Service supports the DROP LIBRARY statement. For more information about the DROP LIBRARY statement, see DROP LIBRARY Statement.

Creating User-Defined Functions

User-defined functions can be created in any programming language that supports COM interfaces.

Parameter and Return Values

A user-defined function can accept any parameter that can be coerced into strings, numbers, or arrays of strings or numbers. User-defined types or object references cannot be used as a parameter. If the parameter data type is explicitly declared as part of the function prototype, such as a double or an integer, Microsoft SQL Server™ 2000 Analysis Services will first coerce values passed into the parameter to the explicitly declared data type. For example, long integer values may be coerced into double precision floating point values if the parameter accepts a Double data type. Date data types are coerced into string representations of a date. PivotTable Service also attempts to coerce strings passed directly into a numeric parameter into numeric values. If coercion fails for any reason, PivotTable Service returns an error condition.

Arrays can also be used as parameters; Analysis Services supports the use of arrays through such functions as **SetToArray**. As with other parameters, if the data type of the array parameter is explicitly declared as part of the function prototype, PivotTable Service will coerce array values into the explicitly declared data type.

If the data type of the array parameter is not explicitly declared or is declared as a variant array, PivotTable Service will also attempt to coerce the elements of the array. However, PivotTable Service handles the coercion of variables in an array a bit differently; the coerced data type is dependent upon the first element of the array, and all other array elements are expected to conform to the same data type. If, for example, the first element in an array is a string, then it is expected that all of the elements in the array are strings, and PivotTable Service will attempt to coerce the other elements into a string data type. If the first element in an array, such as an empty cell, evaluates as empty, then an empty variant is passed to the array parameter instead of a variant array whose first element is empty. If other elements in the array evaluate as empty, the array element is coerced into a zero.

You are recommended to explicitly declare the data types of arrays to be used as parameters in user-defined functions.

Similarly, a user-defined function can return any data type that can be coerced into a number, a string, or a variant. The return values are more restrictive; arrays are not allowed. Additionally, PivotTable Service assumes that if a variant is returned, it contains numeric data. If a string, array, or other non-numeric data is returned through a variant, PivotTable Service returns an error condition for the calculation.

Optional parameters in a function are not supported; PivotTable Service requires all parameters in a user-defined function to be populated.

Functions that return void values (for example, subroutines in Visual Basic) can also be used, but are employed with the CALL keyword. If, for example, you wanted to use the function MyVoidFunction() in an MDX statement, the following syntax would be employed:

```
CALL(MyVoidFunction)
```

Other Considerations

As with any other MDX function, an external function must be resolved before an MDX session can continue; external functions lock MDX sessions while executing. Unless a specific reason exists to halt an MDX session pending user interaction, it is strongly recommended that any user interaction, such as dialog boxes, be discouraged.

External function libraries can duplicate the function names of the Analysis Services function library or other external function libraries. Normally, if an external function library contains a function with the same name as a function in the Analysis Services function library, the Analysis Services function library takes precedence. If two external function libraries contain a function with the same name, the registration order of the external function libraries determines precedence.

However, if you want to override precedence or call a function from a specific external function library, the external function can be preceded by the program ID, delimited with an exclamation point character, as demonstrated here:

```
«ProgramID»!«FunctionName»(«Argument1», «Argument2», ...)
```

If an external function library supports multiple interfaces, the interface ID can also be used to additionally specify the function, as demonstrated here:

```
«ProgramID»!«InterfaceID»!«FunctionName»(«Argument1», «Argum
```

Analysis Services

Using Writebacks

The ability to write information to a write-enabled cube in Multidimensional Expressions (MDX) is called a writeback. Writebacks are supported by two different methods, depending upon the level depth of the member to be changed. Writebacks are supported on server cubes through PivotTable® Service, as described later in this topic. Writebacks to local cubes are not supported.

Lowest-Level Member Writebacks

A lowest-level member is a member in a dimension associated with the lowest defined level of that dimension. For example, in the following diagram, the Products dimension is defined with three levels (not counting the (All) level).

Any writeback to a member at the [Product Name] level is considered a lowest-level writeback, because there are no defined levels below the [Product Name] level.

A separate table is maintained by Microsoft® SQL Server™ 2000 Analysis Services to store data changed by writebacks, and PivotTable Service propagates the data through the affected aggregate members.

For more information about lowest-level writebacks, see Writing a Value Back to a Cell.

Lowest-level writebacks are most commonly used to modify individual lowest-level member data for speculative analysis. If all of the members of a given aggregate are to be modified, it is often easier to use an aggregate-level member writeback.

Aggregate-Level Member Writebacks

An aggregate-level member is any member in a dimension whose value depends upon the value of members related to levels below the aggregate level. For example, in the previous diagram, the [Brand Name] level is an aggregate level because the values for its members depend upon aggregations performed on the [Product Name] level. The [Product Category], too, is an aggregate level, because the values for its members depend upon aggregations created from the [Brand Name] level members.

Aggregate-level writebacks are more difficult to process, because in order to modify an aggregate level, all of the members that are used to construct the values for that aggregate level must be modified. You could individually modify each lowest-level member so that the aggregate level represents the desired value, but for cubes representing thousands, tens of thousands, or more values, this is not a recommended option.

Instead, the UPDATE CUBE statement can be employed, using an allocation. Using one of four different allocation formulas, MDX can distribute the desired aggregate value across all of the lowest level members, in effect handling all of the individual lowest-level writebacks for you. Aggregate-level writebacks can be used only when the values are aggregated using the **Sum** aggregate function.

Aggregate-level writebacks are best used when a correction to an aggregate figure is required affecting all lowest-level members of a particular aggregation. Although lowest-level writebacks can also be used to accomplish this task, the aggregate-level writeback is faster and, because it is treated as a single atomic transaction, ensures that security or formula validation issues will not leave a cube in an inconsistent state.

Note Aggregate-level writebacks may produce imprecise results when integer values are allocated, due to incremental rounding variations.

For more information about aggregate-level member writebacks, see UPDATE CUBE Statement.

Analysis Services

Using DRILLTHROUGH to Retrieve Source Data

The DRILLTHROUGH statement is used in Multidimensional Expressions (MDX) to retrieve a rowset from the source data for a cube cell.

In order to execute a DRILLTHROUGH statement on a cube, drillthrough must be enabled for that cube in the **Drillthrough Options** dialog box. The columns that are returned by a DRILLTHROUGH statement are also specified in this dialog box. (If you are programming with Decision Support Objects (DSO), instead of using the dialog box, you can use the **AllowDrillThrough** and **DrillThroughColumns** properties.) For more information, see Specifying Drillthrough Options.

The following syntax construct describes the DRILLTHROUGH statement:

```
<drillthrough> := DRILLTHROUGH [<Max_Rows>] [<First_Rowset
  < Max_Rows> := MAXROWS <positive number>
  <First_Rowset> := FIRSTROWSET <positive number>
```

The DRILLTHROUGH statement contains a SELECT clause to identify the cube cell for which source data is retrieved. The SELECT clause is identical to an ordinary MDX SELECT statement except that in the SELECT clause only one member can be specified on each axis. If more than one member is specified on an axis, an error occurs.

The <max_rows> syntax specifies the maximum number of the rows in each returned rowset. If the OLE DB provider that is used to connect to the data source does not support DBPROP_MAXROWS, the <max_rows> setting is ignored.

The <first_rowset> syntax identifies the partition whose rowset is returned first.

The following example demonstrates the use of the DRILLTHROUGH statement:

```
DRILLTHROUGH
   SELECT [Warehouse].[All Warehouses].[Canada].[BC] ON ROWS
   [Time].[1998].[Q1] ON COLUMNS,
   [Product].[All Products].[Drink] ON PAGES,
   [Measures].[Units Shipped] ON SECTIONS
   FROM [My Cube]
```

Analysis Services

Understanding Pass Order and Solve Order

Two of the most powerful and, correspondingly, most difficult concepts in Microsoft® SQL Server™ 2000 Analysis Services, solve order and pass order together determine the manner in which a cube is resolved when queries are processed. This topic assumes that you have a basic understanding of cubes, custom members, calculated members, and custom rollups.

Pass Order

When a cube is calculated as the result of a Multidimensional Expressions (MDX) query, it goes through at least one stage of computation, and potentially more stages depending on the use of various calculation-related features, such as custom rollup formulas, custom rollup operators, and calculated cells.

Each stage is referred to as a calculation pass, because the Analysis server makes a complete pass of the calculations applicable for that stage. A calculation pass can be referred to by an ordinal position, called the calculation pass number. The count of calculation passes required to fully compute all the cells of a cube is referred to as the calculation pass depth of the cube.

A cube always has one calculation pass, which retrieves data stored for the cube. Because the ordinal position of the pass number begins at zero, this is always referred to as calculation pass 0. All calculated members and custom members are also calculated on pass 0, and every calculation pass thereafter, with formula precedence within this calculation pass established by the solve order of each calculated member. No other actions are allowed for this calculation pass; calculated cells cannot have calculation pass 0 assigned to their calculation pass number.

If a cube has custom rollup formulas or custom rollup operators, a second calculation pass is performed to handle the computations needed to calculate these features. These features are calculated starting at calculation pass 1, and for every calculation pass thereafter as determined by calculated cells definitions. The calculation pass number cannot be changed for custom rollup formulas or custom rollup operators, because they are calculated on each calculation pass, with formula precedence handled by solve order. However, calculated cells can have calculation pass 1 assigned to their calculation pass number, described in more detail later in this topic.

A cube without calculated cells will have at most two calculation passes. Calculated cells, however, can specify the last calculation pass number on which the calculated cells definition is calculated, and how many passes with which the calculated cells definition is used, providing the ability to create cubes that use two or more calculation passes.

Calculated cells can specify the calculation pass number by using the **Calculation Pass Number** property on the **Advanced** tab of Cube Editor, or by using the CALCULATION_PASS_NUMBER property in MDX statements. Additionally, recursive calculation is allowed by specifying the number of calculation passes to which the calculation formula is recursively applied to the calculation subcube. This feature, accessed through the **Calculation Pass Depth** property on the **Advanced** tab of Cube Editor or by using the CALCULATION_PASS_DEPTH property in MDX statements, can allow highly complex calculations, such as goal-seeking equations, to be employed in a cube. The calculation pass number determines the calculation pass at which evaluation starts and calculation finishes for a calculated cells definition. The calculation pass depth determines how many calculation passes are required to fully compute a calculated cells definition. Only calculated cells will have a calculation pass number higher than 1.

The number of the inclusive range of calculation passes required to fully compute calculated cells can be defined by the formula CALCULATION_PASS_NUMBER to (CALCULATION_PASS_NUMBER - CALCULATION_PASS_DEPTH) + 1, using the cell properties CALCULATION_PASS_NUMBER and CALCULATION_PASS_DEPTH of the calculated cells definition. In other words, if a calculated cells definition has a CALCULATION_PASS_NUMBER of 4 and a CALCULATION_PASS_DEPTH of 3, the calculated cells definition is evaluated in calculation passes 4, 3, and 2, then calculated in calculation passes 2, 3, and 4.

All calculation passes are retained in memory, to facilitate references to previous pass values in calculation formulas. This ability to refer to previous pass values for a given cell allows for complex calculations, such as speculative analysis and goal-seeking formulas, with an increase in performance.

The number of calculation passes required to fully compute all of the cells of a cube is determined by first evaluating all of the custom members, custom rollups, calculated members, and calculated cells. Evaluation is done from highest calculation pass to lowest calculation pass, determined by the CALCULATION_PASS_NUMBER property, in order to accurately determine formula precedence across calculation passes. The order is then reversed when calculating the calculation passes, by calculating from lowest to highest. Essentially, each calculation pass is treated as a nested calculation, with the lowest calculation pass being the most nested.

The following table illustrates the effects of calculation pass number and calculation pass depth on a sample cube. The sample cube contains four calculations:

- A calculated member, shaded in dark gray, with a SOLVE_ORDER of 1.

- A custom rollup formula, shaded in dark blue, with a SOLVE_ORDER of 2.

- A calculated cells definition, shaded in green, with a CALCULATION_PASS_NUMBER of 2, a CALCULATION_PASS_DEPTH of 1, and a SOLVE_ORDER of 1.

- A calculated cells definition, shaded in red, with a CALCULATION_PASS_NUMBER of 3, a CALCULATION_PASS_DEPTH of 2, and a SOLVE_ORDER of 2.

Pass diagram	Pass description
	Calculation Pass 3 Because the CALCULATION_PASS_DEPTH of the calculated cells definition shaded with red is 2, the cells are recursively calculated again, using the values derived from the previous calculation pass. The calculated member and

	custom rollup formula are also calculated again on this pass. **Calculation Pass 2** The calculations for both calculated cells definitions start here, based on the evaluation of CALCULATION_PASS_NUMBER and CALCULATION_PASS_DEPTH. The calculated member and custom rollup formula are also calculated again on this pass. **Calculation Pass 1** All custom rollup formulas and custom rollup operators start calculation on pass 1. **Calculation Pass 0** Data is retrieved from sources. Calculated and custom members are calculated. No other calculations can be performed at this point.

Color-coded arrows show evaluation and calculation order for the various calculations in the diagram shown in the previous table. In cubes with multiple calculations, some of the calculations can overlap. When this occurs, the solve order of the overlapping calculations is used to resolve formula precedence, but only within a given pass. If a solve order is specified for a calculated cells definition that encompasses multiple passes (that is, the CALCULATION_PASS_DEPTH is greater than 1), the solve order is applied to each pass to resolve formula precedence. Because solve order is applied on each calculation pass, overlapping calculated cells definitions can generate different values for cells that may be involved in custom rollup or calculated member resolution. A more detailed discussion of solve order is given later in this topic.

Recursive calculations and goal-seeking calculations can make use of values obtained in previous passes through the use of the **CalculationPassValue** and **CalculationCurrentPass** functions in MDX. The **CalculationCurrentPass** function provides the current calculation pass number, and the **CalculationPassValue**, given an MDX expression and a calculation pass number, evaluates the MDX expression within the specified calculation pass number and returns the result.

Solve Order

Within a single pass, solve order determines two things: the order in which dimensions, members, calculated members, custom rollups, and calculated cells are evaluated, and the order in which they are calculated. The member with the highest solve order is evaluated first, but calculated last. This is similar in behavior to any other nested operation: the outermost operation cannot complete until the innermost operation is completed, but the outermost operation is evaluated first in order to determine that the innermost operation must be completed before the outermost can be completed. The lower the solve order, the more nested the member in terms of evaluation and calculation, with the member having the highest solve order occupying the outermost position.

In cubes with dimensions that contain custom members, custom rollup formulas, calculated members, and calculated cells, the solve order determines the order in which various calculations are evaluated. The highest solve order is always evaluated first, then the next highest, and so on.

The order in which they are calculated, however, is reversed. The lowest solve order is calculated first, then the next lowest, and so on. Solve order essentially nests formulas, and as with any nested formula, the outermost formulas are evaluated first, but calculated last. The innermost formulas are evaluated last, but calculated first, because the outermost formulas may depend on the values produced by the innermost formulas for their calculation.

Although measures are usually treated as another dimension, they are always evaluated last and calculated first for solve order purposes. In other words, measures are always treated as having the lowest possible solve order.

In Cube Editor, the solve order for calculated members and calculated cells can be changed by altering the **Solve Order** property in the **Advanced** tab of the properties pane. The solve order for dimensions can be changed by reordering the positions of the dimensions within the tree pane. For more information about using Cube Editor, see Cube Editor - Schema View.

In MDX the **SOLVE_ORDER** member property can be used when creating or changing calculated members and calculated cells

Solve order directly affects the results generated by the calculation of dimensions and members in this fashion. The following diagram describes the behavior of two dimensions, each with a calculated member, that intersect at a cell. Two examples are presented in the following diagram, with different solve orders.

In the first example, Dimension 2 has a higher solve order. So, the intersection is evaluated using the formula for Dimension 2. However, in order to provide data for this formula, the calculated member in Dimension 1 must be evaluated and calculated. So, the formula for Dimension 1 is calculated to provide the values needed to calculate the formula for Dimension 2. Then, the formula for Dimension 2 is calculated and the result is placed in the cell.

In the second example, Dimension 1 has a higher solve order. The cell at the intersection is evaluated using the formula for Dimension 1. As Dimension 2 has a lower solve order, it is evaluated next, then calculated first and the values provided to the formula for Dimension 1. Then, the formula for Dimension 1 is calculated and the result is placed in the cell.

To further demonstrate the potential complexities of solve order, a series of example MDX queries is presented here.

For the first example, you are interested in seeing the difference in income and expenses for each half of the year. You would then construct a simple MDX query similar to the following example:

```
WITH
MEMBER [Time].[Year Difference] AS
   '[Time].[2nd half] - [Time].[1st half]
SELECT
   { [Money].[Income], [Money].[Expenses] } ON COLUMNS,
   { [Time].[1st half], [Time].[2nd half], [Time].[Year Difference] } ON
FROM TestCube
```

This MDX query would produce a result set similar to the following table, with the calculated member shaded.

	Income	Expenses
1st half	5000	4200
2nd half	8000	7000
Year Difference	3000	2800

For this query, solve order is not an issue, assuming the cube does not use any calculated members, because there is only one calculated member in the query.

Now, for the second example, you are interested in seeing the percentage of net income after expenses for each half of the year, using the following MDX query:

```
WITH
MEMBER [Money].[Net Income] AS
   '([Money].[Income]  [Money].[Expenses]) / [Money].[Income]'
SELECT
   { [Money].[Income], [Money].[Expenses], [Money].[Net Income] }
   { [Time].[1st half], [Time].[2nd half] } ON ROWS
FROM TestCube
```

This MDX query would produce a slightly different result set, similar to the following table, with the calculated member shaded.

	Income	Expenses	Net Income
1st half	5000	4200	0.16
2nd half	8000	7000	0.125

This MDX query, like the previous one, does not have any solve order complications, because it also has only a single calculated member. Notice the placement of the calculated member in the result dataset of this example, as well as in the previous example. The first MDX query example uses a calculated member as part of the ROWS axis dimension, but this query example uses a calculated member as part of the COLUMNS axis dimension. This placement becomes important in the next example, which combines the two calculated members in a single MDX query.

Finally, you decide you want to combine both of the previous examples into a single MDX query. In this case, solve order becomes important. Take, for example, the first attempt at this combination in the following MDX query:

```
WITH
MEMBER [Time].[Year Difference] AS
   '[Time].[2nd half] - [Time].[1st half],
   SOLVE_ORDER = 1
MEMBER [Money].[Net Income] AS
   '([Money].[Income] - [Money].[Expenses]) / [Money].[Income]',
   SOLVE_ORDER = 2
SELECT
   { [Money].[Income], [Money].[Expenses], [Money].[Net Income] }
   { [Time].[1st half], [Time].[2nd half], [Time].[Year Difference] } ON
FROM TestCube
```

The two calculated members, Year Difference and Net Income, intersect at a single cell in the result dataset of the MDX query example. The only way to determine how this cell will be evaluated is by the solve order. The formulas used to construct this cell will produce different results depending upon the solve order of the two calculated members.

The SOLVE_ORDER keyword is be used to specify the solve order of calculated members in an MDX query or the CREATE MEMBER command. If the solve order is not specified, it defaults to zero. In that case, the order of the dimensions in the cube whose context is specified in the MDX query is used to determine the solve order, following the rules listed earlier in this topic. This also applies to calculated members in different dimensions that are assigned the same SOLVE_ORDER value.

The integer values used with the SOLVE_ORDER keyword are relative; the value simply tells MDX to calculate a member based on values derived from calculating members with a higher value. If a calculated member is defined without the SOLVE_ORDER keyword, its default value is zero. The specified values do not need to start at zero, nor do they need to be consecutive.

In the MDX query example, Net Income has the highest solve order, so the cell in question is evaluated using the Net Income formula. But the Net Income formula is calculated last; in order to calculate Net Income, the next calculated member, Year Difference, must be calculated first so that Net Income can use the results of that calculated member to perform its own calculation.

The results of this nested calculation can be viewed in the following table.

	Income	Expenses	Net Income
1st half	5000	4200	0.16
2nd half	8000	7000	0.125
Year Difference	3000	2800	0.066

As you can see, the result in the shared cell is based on the formula for Net Income; in other words, it was calculated with the Year Difference data, producing the following formula (the result is rounded for clarity):

((8000 - 5000) - (7000 - 4200)) / (8000 - 5000) = 0.066

or

(3000 - 2800) / 3000 = 0.066

The result in the shared cell, however, is calculated differently if the solve orders for the calculated members in the MDX query are switched, as demonstrated here:

```
WITH
MEMBER [Time].[Year Difference] AS
  '[Time].[2nd half] - [Time].[1st half],
  SOLVE_ORDER = 2
MEMBER [Money].[Net Income] AS
  '([Money].[Income] - [Money].[Expenses]) / [Money].[Income]',
  SOLVE_ORDER = 1
SELECT
  { [Money].[Income], [Money].[Expenses], [Money].[Net Income] } 
  { [Time].[1st half], [Time].[2nd half], [Time].[Year Difference] } ON
```

FROM TestCube

As the order of the calculated members has been switched, the Year Difference formula is used to evaluate the cell. The Net Income calculated member is resolved first, and then the Year Difference calculated member is resolved, producing a strikingly different result as shown in the following table.

	Income	**Expenses**	**Net Income**
1st half	5000	4200	0.16
2nd half	8000	7000	0.125
Year Difference	3000	2800	-0.035

Because it uses the Year Difference formula with the Net Income data, the formula for the shared cell resembles the following calculation:

((8000 - 7000) / 8000) - ((5000 - 4200) / 5000) = -0.035

Or

0.125 - 0.16 = -0.035

Changing Solve Order Values

Solve order values can range from -8181 to 65535. It is highly recommended that you use only positive integers when setting solve order values. Certain calculations reside at specific solve orders, as listed in the following table. The solve order for a pass can become unpredictable if these values are used by other calculations.

Calculation	Solve order
Calculated cell formula "dirtiness"	-6143
Custom rollup formulas (if not otherwise specified)	-5119
Virtual dimensions created with earlier versions of Analysis Services	-4097
Visual totals calculation	-4096
All other calculations (if not otherwise specified)	0

For example, changing the solve order for a calculated cells definition below the default custom rollup formula value of -5119 causes the calculated cells definition to be calculated before the custom rollup formulas; this can produce incorrect results.

In the case of multiple calculations having the same solve order, the following formula precedence is used:

1. Calculated cells

2. Custom rollup formulas

3. Custom and calculated members

4. All other calculations

Calculated cells take precedence over all other calculations in the case of solve order conflict. If multiple calculations occur within the same category, the declaration order of the calculation is used. For example, if two calculated cells definitions have the same solve order for the same calculation pass, the declaration order determines which is evaluated first.

Additional Considerations

The combination of pass order and solve order can be a very complex issue to deal with, especially in cubes with a high number of dimensions involving calculated member, custom rollup formulas, or calculated cells. When MDX evaluates an MDX query, the solve order values for everything involved within a given pass, including the dimensions of the cube specified in the MDX query, are taken into account.

When a query with calculated members is executed against a cube with calculated members, for example, the solve orders for both the query and the cube are evaluated as if the query were part of the cube; it is executed within the context of a cube. Because it can be difficult to review the solve order of the dimensions on a cube, it can be challenging to ensure that the solve order for calculated members in a complex MDX query are correctly handled within the context of a cube.

Also, as the solve order for the dimensions on a cube can be changed from Cube Editor, MDX queries can be affected; a once-working MDX query can return unexpected results because the solve order of the cube on which context the MDX query executes is changed.

See Also

CalculationCurrentPass

CalculationPassValue

Cube Editor - Schema View

CREATE MEMBER Statement

Analysis Services

Effective MDX

This topic provides information on more effective uses of Multidimensional Expressions (MDX) functions in various scenarios. The topics covered are listed in the following table.

Topic	Description
Comments in MDX	Describes the use of comments in MDX statements.
Working with Empty Cells	Explains the behavior and functions associated with empty cells in MDX.
Creating a Cell Within the Context of a Cube	Covers the creation of an MDX query that returns a single cell.
Working with the RollupChildren Function	Discusses the impact of the **RollupChildren** function on analysis of multidimensional data.
WHERE Clause Overrides	Details how and when the WHERE clause of an MDX query can override the resolution of individual set, member, tuple, or numeric functions in an MDX statement.

Analysis Services

Comments in MDX

Statements in Multidimensional Expressions (MDX) can contain user-readable comments that are ignored when the commands are processed. The three different character sets that indicate comments are outlined in the following table.

Characters	Description
//	C++-style forward slashes. All text between the forward slashes and the end of the same line is ignored.
--	SQL-style hyphens. All text between the dashes and the end of the same line is ignored.
/*...*/	C-style forward slash and asterisk pairs. All text between the opening forward slash and asterisk and the closing asterisk and backward slash is ignored. This type of comment can span multiple lines.

The following example shows the use of comments in an MDX command:

```
/* Using this query to view
   info about units shipped
   and units ordered  */

WITH MEMBER [Measures].[ShippingPercent] AS
'-- Returns [Units Shipped] over [Units Ordered] as a percent value
Measures.[Units Shipped] / Measures.[Units Ordered]',
FORMAT_STRING = 'Percent'

SELECT
   { [Measures].[Units Shipped], [Measures].[Units Ordered], [Measure
// The next command specifies nonempty members only
   NON EMPTY [Store].[Store Name].Members ON ROWS
FROM Warehouse   -- Pulled from the Warehouse cube
```

Comments are recommended in complex or difficult to understand MDX queries, because they add information without incurring performance penalties.

Analysis Services

Working with Empty Cells

Empty cells occur in Multidimensional Expressions (MDX) statements when data for the intersection of two or more dimensions does not exist. For example, the following MDX query example produces many empty cells:

```
SELECT
  {[Store].[Store Name].Members} ON COLUMNS,
  {[Product].[Excellent Diet Cola]} ON ROWS
FROM Sales
WHERE [Measures].[Unit Sales]
```

The product, Excellent Diet Cola, is not sold in all stores. For the stores that sell the product, the Unit Sales measure will contain a numeric value. For the stores that do not sell the product, however, an empty cell will be displayed.

Empty cells affect the evaluation of value expressions and search conditions. To understand why this is so, note that a value expression is composed of value expression primaries. One of the value expression primaries is `<tuple>[.VALUE]`, which returns the value of a cell in the cube (some of whose coordinates are specified explicitly by `<tuple>`, and others that are available implicitly from the context of the MDX statement). This cell can be an empty cell. Empty cells affect expression evaluation in the following three cases:

- With numeric value expressions. In a numeric value expression, this value can be added, subtracted, multiplied, or divided by other values. It can also appear as the parameter of any function that has a <numeric_value_expression> argument.

- With string value expressions. In a string value expression, this value can be concatenated to another string.

- With search conditions composed of Boolean primaries. A Boolean primary is of the following form:
  ```
  <boolean_primary> ::= <value_expression> <comp_op> <valu
  ```

A value expression will be made up of the value expression primary, and this will lead to the first two cases described listed earlier.

Empty Cell Evaluation

MDX specifically identifies an empty cell by defining a special empty cell value that is present in an empty cell. The empty cell value is evaluated as follows:

- The function **IsEmpty**(<value_expression>) returns TRUE if <value_expression> is the empty cell value. Otherwise it returns FALSE.
- When the empty cell value is an operand for any of the numeric operators (+, -, *, /), it behaves like the number zero.
- When the empty cell value is an operand for the string concatenation operator (||), it behaves like the empty string.
- When the empty cell value is an operand for any of the comparison operators (=. <>, >=, <=, >, <), it behaves like the number zero or the empty string, depending on whether the data type of the other operand is numeric or string, respectively.
- When collating numeric values, the empty cell value collates in the same place as zero. Between the empty cell value and zero, empty collates before zero.
- When collating string values, the empty cell value collates in the same place as the empty string. Between the empty cell value and the empty string, the empty cell value collates before an empty string.

Empty cells can be handled in a variety of ways; the easiest is to simply remove them from consideration. However, because this is not always practical in MDX, functions have been provided to deal with empty cells.

NON EMPTY Keyword

The easiest way to remove empty cells from consideration is to use the NON EMPTY keyword in an MDX query. The following example is the same MDX query example discussed earlier in this topic, but using the NON EMPTY keyword.

```
SELECT
   NON EMPTY {[Store].[Store Name].Members} ON COLUMNS,
   {[Product].[Excellent Diet Cola]} ON ROWS
FROM Sales
WHERE [Measures].[Unit Sales]
```

All of the stores in the first axis dimension that do not have values for the unit sales of the product are excluded from the result dataset. The empty tuples are screened out of the result dataset of the MDX query.

It is important to note that this function screens out empty tuples, not individual empty cells. Because of this, empty cells can appear in a result dataset even when the NON EMPTY keyword is used. For example, suppose you want to examine the unit sales for two different products in 1997 for each store. The following MDX query example uses the NON EMPTY keyword to screen out empty tuples:

```
SELECT
   NON EMPTY CROSSJOIN ({[Product].[Excellent Diet Cola], [Proc
   NON EMPTY {[Store].[Store Name].Members} ON ROWS
FROM Sales
WHERE [Measures].[Unit Sales]
```

However, the result dataset resembles the following table.

	Excellent Diet Soda	**Fabulous Diet Soda**
	1997	**1997**
Store 6	20.00	11.00
Store 7	25.00	6.00
Store 24	11.00	19.00

Store 11	36.00	32.00
Store 13	25.00	22.00
Store 2	2.00	
Store 3	23.00	16.00
Store 15	14.00	17.00
Store 16		13.00
Store 17	22.00	12.00
Store 22	2.00	
Store 23	4.00	5.00

The result dataset still shows three empty cells, despite the presence of the NON EMPTY keyword. The tuples created by the MDX query may contain empty cells, but the tuples themselves are not empty. For example, in the preceding result dataset, though Store 22 did not sell any of the Fabulous Diet Soda product in 1997, it did sell some of the Excellent Diet Soda product in 1997. So, the tuple created by the CROSSJOIN command does contain a member that does not evaluate to an empty cell; therefore the tuple is not considered empty and is not screened out.

For more information about the use of NON EMPTY in MDX SELECT statements, see SELECT Statement.

CoalesceEmpty Function

This MDX function returns the first nonempty value in a list of values. It is useful when you want to replace empty cell values with another numeric or string expression.

The **CoalesceEmpty** function allows you to evaluate a series of value expressions from left to right. The first value expression in the series that does not evaluate to the empty cell value is returned. For example, the following MDX query modifies the previous MDX query example to replace all of the empty cell values in the Unit Sales measure with zero:

```
WITH MEMBER [Measures].[NonEmptyUnitSales] AS
   'CoalesceEmpty(Measures.[Unit Sales], 0)'
```

```
SELECT
   NON EMPTY CROSSJOIN ({[Product].[Excellent Diet Cola], [Proc
   NON EMPTY {[Store].[Store Name].Members} ON ROWS
FROM Sales
WHERE [Measures].[NonEmptyUnitSales]
```

The following table demonstrates the result dataset returned by the MDX query example.

	Excellent Diet Soda	**Fabulous Diet Soda**
	1997	**1997**
Store 19	0	0
Store 20	0	0
Store 9	0	0
Store 21	0	0
Store 1	0	0
Store 5	0	0
Store 10	0	0
Store 8	0	0
Store 4	0	0
Store 12	0	0
Store 18	0	0
HQ	0	0
Store 6	20.00	11.00
Store 7	25.00	6.00
Store 24	11.00	19.00
Store 11	36.00	32.00
Store 13	25.00	22.00
Store 2	2.00	0
Store 3	23.00	16.00
Store 15	14.00	17.00
Store 16	0	13.00

Store 17	22.00	12.00
Store 22	2.00	0
Store 23	4.00	5.00

The values of the calculated member NonEmptyUnitSales were determined by the **CoalesceEmpty** function. If the Unit Sales value evaluated to a nonempty cell, the first value in the **CoalesceEmpty** statement was returned. If the [Unit Sales] value evaluated to an empty cell value, the second value in the **CoalesceEmpty** statement was returned. Because the **CoalesceEmpty** function replaced all of the empty cell values with zero, the NON EMPTY keyword has nothing to screen out, so all of the tuples in the query were valid and were presented in the result dataset.

Other Functions

The way that other functions (especially calculation functions) deal with empty cells depends on the capabilities and options that are available to those functions. Functions such as **Count** and **Avg** evaluate a count of cells, but whether or not to evaluate an empty cell by this type of function should be given careful thought. In practice, it is sometimes preferable to count the number of empty cells. For example, when the number of sales representatives is counted as part of a performance evaluation query, all sales representatives should be included in the count whether or not they sold anything. In this case, each no-sale results in an empty cell. However, there are other situations in which empty cells should not be counted, such as when getting the average of sales over a certain domain. In this case, counting the no-sale cells would inaccurately decrease the average.

Some MDX functions in which empty cells may change the outcome allow for the inclusion or exclusion of empty cells as part of their calculation. **Count**, for example, supports the use of INCLUDEEMPTY and EXCLUDEEMPTY flags to handle the inclusion or exclusion of empty cells, respectively, while counting.

Analysis Services

Creating a Cell Within the Context of a Cube

For certain applications, you may want to return data for a single cell within a cube. For example, executives might have a decision support application written in Microsoft® Excel that uses data from a multidimensional data store. Suppose that when the application starts each day, the executives want to view, at the top of the application's main window, the quarter-to-date worldwide sales for the current year across all products and customers.

The solution is to create a dataset for which all dimensions are slicer dimensions. The Multidimensional Expressions (MDX) statement for doing this takes the following form:

```
SELECT FROM cube_name WHERE slicer_specification
```

This results in a dataset with one cell. Because no axis dimensions are specified, the slicer specification focuses on the desired point in the entire cube.

In this case, where there are no axes and hence only one cell, the following conditions apply:

- The IMDDataset::GetAxisInfo method returns 0 for **pcAxes* and a null pointer in *prgAxisInfo.
- The axis rowsets for all axes will be empty, except for the axis MDAXIS_SLICERS. The axis for the slicer dimension will contain information on the slicer conditions that created the single cell.
- The single cell can be addressed by using the cell ordinal 0.

Analysis Services

Working with the RollupChildren Function

The use of the **RollupChildren** function in Multidimensional Expressions (MDX) statements is simple to explain, but the impact of this function on MDX queries can be wide-ranging.

The **RollupChildren** function rolls up the children of a member, applying a different unary operator to each child, and returns the value of this rollup as a number. The unary operator used can be supplied by a member property associated with the child member, or it can be a string expression provided directly to the function.

The impact of the **RollupChildren** function occurs in MDX queries designed to perform selective analysis on existing cube data. For example, the following table contains a list of child members for the Net Sales parent member, with their unary operators (represented by the UNARY_OPERATOR member property) shown in parentheses.

Parent member	Child member
Net Sales	Domestic Sales (+) Domestic Returns (-) Foreign Sales (+) Foreign Returns (-)

The Net Sales parent member currently provides a total of net sales minus the gross domestic and foreign sales values, with the domestic and foreign returns subtracted as part of the rollup.

Now, if you want to provide a quick and easy forecast of domestic and foreign gross sales plus 10%, ignoring the domestic and foreign returns, there are two ways to perform this action using the **RollupChildren** function.

Custom Member Properties

If this is to be a commonly performed operation, one method is to create a member property that stores the operator to be used for each child for a given function. For example, a member property called SALES_OPERATOR is created, and the following unary operators are assigned to it, as shown in the following table.

Parent member	Child member
Net Sales	Domestic Sales (+) Domestic Returns (~) Foreign Sales (+) Foreign Returns (~)

With this new member property, the following MDX statement performs the gross sales estimate operation quickly and efficiently:

```
RollupChildren([Net Sales], [Net Sales].CurrentMember.Properties("S
```

When the function is called, the value of each child is applied to a total using the operator stored in the member property. The following table displays valid unary operators and describes the expected result.

Operator	Result
+	total = total + current child
-	total = total - current child
*	total = total * current child
/	total = total / current child
~	Child is not used in the rollup. Its value is ignored.

The tilde (~) unary operator indicates that this member is to be ignored when generating rollups totals. The members for domestic and foreign returns are ignored and the rollup total returned by the **RollupChildren** function is multiplied by 1.1.

IIf Function

However, if the example operation is not commonplace or if it applies only to one MDX query, then the **IIf** function can be used with the **RollupChildren** function to provide the same result. The following MDX query provides the same result as the earlier MDX example, but does so without resorting to the use of a custom member property:

```
RollupChildren([Net Sales], IIf([Net Sales].CurrentMember.Properties
```

The MDX statement checks the unary operator of the child member; if it is used for subtraction (as with the domestic and foreign returns members), the tilde (~) unary operator is substituted by the **IIf** function. Otherwise, the unary operator of the child member is used. Finally, the returned rollup total is then multiplied by 1.1 to provide the domestic and foreign gross sales forecast value.

See Also

RollupChildren

Analysis Services

WHERE Clause Overrides

Each individual set, member, tuple, or numeric function in a Multidimensional Expressions (MDX) statement always executes in the larger context of the entire statement. For example, consider the FILTER function in the following expression:

```
SELECT FILTER(SalesRep.MEMBERS, [1996].VALUE > 500) ON C
   Quarters.MEMBERS ON ROWS
FROM SalesCube
WHERE ([Geography].[All], [Products].[All], [1996], Sales)
```

The second argument of FILTER, "[1996].VALUE", does not contain enough information by itself. Six coordinates are needed, one from each of the six dimensions, to determine VALUE. The argument contains only one coordinate, from the Years dimension. In such a case, the other coordinates are obtained by looking at the following, in order:

1. The rest of the axis specification. This yields (in the preceding example) the coordinate of the SalesRep dimension because the FILTER function iterates through each member of the SalesRep dimension.

2. The slicer condition (WHERE clause) and the coordinates for the slicer dimension. This yields the coordinates for the Geography, Products, and Measures dimensions as (respectively) Geography.[All], Products.[All], and Measures.Sales.

3. The default member for dimensions that appear neither on the axis nor on the slicer. Thus the default members are picked for the Quarters dimension.

A special case arises when a coordinate is specified both in the WHERE clause and within the expression. For example, suppose an application calls for a dataset that, on the COLUMNS axis, contains 1996 budgeted sales for all the states in the United States that had more than 500 units of ActualSales in 1995 and that, on the ROWS axis, contains the Quarters. The following statement can create this dataset:

```
SELECT FILTER({USA.CHILDREN}, ([1995], ActualSales) > 500) (
   Quarters.MEMBERS ON ROWS
FROM SalesCube
WHERE ([1996], BudgetedSales, [Products].[All], [SalesRep].[All])
```

As the FILTER function is evaluated for each state in the United States, it already has the coordinates ([1996], BudgetedSales) from the WHERE clause. However, it receives the coordinates ([1995], ActualSales) from the FILTER function. To avoid potential conflict, the argument of the FILTER function takes precedence. In general, any coordinates obtained from the WHERE clause are overridden by coordinates that are specified within an axis specification.

Analysis Services

MDX Functions in Analysis Services

Microsoft® SQL Server™ 2000 Analysis Services provides for the use of functions in Multidimensional Expressions (MDX) syntax. Functions can be used in any valid MDX statement, and are often used in queries, calculated members, and custom rollup definitions. There are three types of functions in MDX, and each is described in a separate topic. The following table lists, describes, and provides links to each topic.

Topic	Description
MDX Function Reference	Provides a list of functions intrinsic to MDX, including syntax and examples.
Registered Function Libraries	Describes the use of external function libraries, such as the Microsoft Excel worksheet library and the Microsoft Visual Basic® for Applications Expression Services library, in MDX expressions.

Analysis Services

MDX Function Reference

This topic provides information about the Multidimensional Expressions (MDX) functions included with Microsoft® SQL Server™ 2000 Analysis Services. You can use the MDX Function List to find functions by their category of return value, or you can select a function by name from the alphabetical list in the table of contents.

See Also

MDX Function List

MDX Syntax Conventions

The diagrams for Multidimensional Expressions (MDX) syntax in the MDX Function Reference use these conventions.

Convention	**Usage**
[ ] (brackets)	Optional syntax items. Do not type the brackets.
| (vertical bar)	Separating syntax items within brackets or braces. You can choose only one of the items.
« » (guillemets)	User-supplied parameters of MDX syntax. Do not type the guillemets.
[,...]	Indicating that the preceding item can be repeated any number of times. The items are separated by commas.

MDX Function List

This topic contains lists of the Multidimensional Expressions (MDX) functions in Microsoft® SQL Server™ 2000 Analysis Services. You can use these lists to find functions by their category of return value, or you can select a function by name from the alphabetical list in the table of contents.

Samples Used in Examples

For many expression examples in the following topics, SampleSet is defined as:

{USA, Buffalo, France, NYC, London, California, LA, Nice, UK, Pari

The following table lists sales data for each member of the set.

Location	**1995 sales**	**1996 sales**
UK	1900	1700
London	250	300
France	2500	2500
Paris	365	250
Nice	27	100
USA	5000	6500
Boston	900	1100
Buffalo	300	200
California	2000	3500
Los Angeles	500	900

MDX Function Groups

The following tables list the MDX functions grouped by their return value categories. You can use the links in the tables to jump to the function reference topics.

Array Functions

Function	Description
SetToArray	Converts one or more sets to an array for use in a user-defined function.

Dimension, Hierarchy, and Level Functions

Dimension Functions

Function	Description
Dimension	Returns the dimension that contains a specified hierarchy, level, or member.
Dimensions	Returns the dimension whose zero-based position within the cube is specified by a numeric expression or whose name is specified by a string.

Hierarchy Functions

Function	Description
Hierarchy	Returns the hierarchy of a level or member.

Level Functions

Function	Description
Level	Returns the level of a member.
Levels	Returns the level whose position in a dimension is specified by a numeric expression or whose name is specified by a string expression.

Logical Functions

Function	Description
Is	Returns True if two compared objects are equivalent, False otherwise.
IsAncestor	Determines whether a specified member is an ancestor of another specified member.
IsEmpty	Determines whether an expression evaluates to the empty cell value.
IsGeneration	Determines whether a specified member is in a specified generation.
IsLeaf	Determines whether a specified member is a leaf member.
IsSibling	Determines whether a specified member is a sibling of another specified member.

Member Functions

Function	Description
Ancestor	Returns the ancestor of a member at a specified level or at a specified distance from the member.
ClosingPeriod	Returns the last sibling among the descendants of a member at a level.
Cousin	Returns the member with the same relative position under a member as the member specified.
CurrentMember	Returns the current member along a dimension during an iteration.
DataMember	Returns the system-generated data member associated with a nonleaf member.
DefaultMember	Returns the default member of a dimension or hierarchy.
FirstChild	Returns the first child of a member.
FirstSibling	Returns the first child of the parent of a member.
Ignore	Reserved.
Item	Returns a member from a tuple.
Lag	Returns a member prior to the specified member along the member's dimension.

LastChild	Returns the last child of a member.
LastSibling	Returns the last child of the parent of a member.
Lead	Returns a member further along the specified member's dimension.
LinkMember	Returns a hierarchized member.
Members	Returns the member whose name is specified by a string expression.
NextMember	Returns the next member in the level that contains a specified member.
OpeningPeriod	Returns the first sibling among the descendants of a member at a level.
ParallelPeriod	Returns a member from a prior period in the same relative position as a specified member.
Parent	Returns the parent of a member.
PrevMember	Returns the previous member in the level that contains a specified member.
StrToMember	Returns a member based on a string expression.
ValidMeasure	Returns a valid measure in a virtual cube by forcing inapplicable dimensions to their top level.

Numeric Functions

Function	**Description**
Aggregate	Returns a calculated value using the appropriate aggregate function, based on the context of the query.
Avg	Returns the average value of a numeric expression evaluated over a set.
CalculationCurrentPass	Returns the current calculation pass of a cube for the current query context.
CalculationPassValue	Returns the value of an MDX expression evaluated over a specified calculation pass of the current cube.

CoalesceEmpty	Coalesces an empty cell value to a number or a string.
Correlation	Returns the correlation of two series evaluated over a set.
Count	Returns the number of dimensions in a cube, the number of levels in a dimension, the number of cells in a set, or the number of dimensions in a tuple.
Covariance	Returns the population covariance of two series evaluated over a set, using the biased population formula.
CovarianceN	Returns the sample covariance of two series evaluated over a set, using the unbiased population formula.
DistinctCount	Returns the count of tuples in a set, excluding duplicate tuples.
IIf	Returns one of two numeric or string values determined by a logical test.
LinRegIntercept	Calculates the linear regression of a set and returns the value of b in the regression line y = ax + b.
LinRegPoint	Calculates the linear regression of a set and returns the value of y in the regression line y = ax + b.
LinRegR2	Calculates the linear regression of a set and returns R^2 (the coefficient of determination).
LinRegSlope	Calculates the linear regression of a set and returns the value of a in the regression line y = ax + b.
LinRegVariance	Calculates the linear regression of a set and returns the variance associated with the regression line y = ax + b.
LookupCube	Returns the value of an MDX expression evaluated over another specified cube in the same database.
Max	Returns the maximum value of a numeric expression evaluated over a set.
Median	Returns the median value of a numeric expression evaluated over a set.

Min	Returns the minimum value of a numeric expression evaluated over a set.
Ordinal	Returns the zero-based ordinal value associated with a level.
Predict	Evaluates the string expression within the data mining model specified within the current coordinates.
Rank	Returns the one-based rank of a tuple in a set.
RollupChildren	Scans the children of the member parameter and applies the string expression operator to their evaluated value.
Stddev	Alias for **Stdev**.
StddevP	Alias for **StdevP**.
Stdev	Returns the sample standard deviation of a numeric expression evaluated over a set, using the unbiased population formula.
StdevP	Returns the population standard deviation of a numeric expression evaluated over a set, using the biased population formula.
StrToValue	Returns a value based on a string expression.
Sum	Returns the sum of a numeric expression evaluated over a set.
Value	Returns the value of a measure.
Var	Returns the sample variance of a numeric expression evaluated over a set, using the unbiased population formula.
Variance	Alias for **Var**.
VarianceP	Alias for **VarP**.
VarP	Returns the population variance of a numeric expression evaluated over a set, using the biased population formula.

Other Functions

Function	Description
Call	Executes the string expression containing a user-defined function.

Set Functions

Function	Description
AddCalculatedMembers	Adds calculated members to a set.
AllMembers	Returns a set containing all members of a specified dimension or level, including calculated members.
Ancestors	Returns all the ancestors of a member at a specified distance.
Ascendants	Returns the set of the ascendants of the member, including the member itself.
Axis	Returns the set associated with the main axis.
BottomCount	Returns a specified number of items from the bottom of a set, optionally ordering the set first.
BottomPercent	Sorts a set and returns the bottom *n* elements whose cumulative total is at least a specified percentage.
BottomSum	Sorts a set and returns the bottom *n* elements whose cumulative total is at least a specified value.
Children	Returns the children of a member.
Crossjoin	Returns the cross product of two sets.
Descendants	Returns the set of descendants of a member at a specified level or at a specified distance from a member, optionally including or excluding descendants in other levels.
Distinct	Eliminates duplicate tuples from a set.
DrilldownLevel	Drills down the members of a set, at a specified level, to one level below.

	Alternatively, drills down on a specified dimension in the set.
DrilldownLevelBottom	Drills down the bottom *n* members of a set, at a specified level, to one level below.
DrilldownLevelTop	Drills down the top *n* members of a set, at a specified level, to one level below.
DrilldownMember	Drills down the members in a set that are present in a second specified set.
DrilldownMemberBottom	Similar to **DrilldownMember**, except that it includes only the bottom *n* children.
DrilldownMemberTop	Similar to **DrilldownMember**, except that it includes only the top *n* children.
DrillupLevel	Drills up the members of a set that are below a specified level.
DrillupMember	Drills up the members in a set that are present in a second specified set.
Except	Finds the difference between two sets, optionally retaining duplicates.
Extract	Returns a set of tuples from extracted dimension elements. The opposite of **Crossjoin**.
Filter	Returns the set resulting from filtering a set based on a search condition.
Generate	Applies a set to each member of another set and joins the resulting sets by union.
Head	Returns the first specified number of elements in a set.
Hierarchize	Orders the members of a set in a hierarchy.
Intersect	Returns the intersection of two input sets, optionally retaining duplicates.
LastPeriods	Returns a set of members prior to and including a specified member.
Members	Returns the set of all members in a dimension, hierarchy, or level.
Mtd	A shortcut function for the **PeriodsToDate**

	function that specifies the level to be Month.
NameToSet	Returns a set containing a single member based on a string expression containing a member name.
NonEmptyCrossjoin	Returns the cross product of two or more sets, excluding empty members.
Order	Arranges members of a set, optionally preserving or breaking the hierarchy.
PeriodsToDate	Returns a set of periods (members) from a specified level starting with the first period and ending with a specified member.
Qtd	A shortcut function for the **PeriodsToDate** function that specifies the level to be Quarter.
Siblings	Returns the siblings of a member, including the member itself.
StripCalculatedMembers	Removes calculated members from a set.
StrToSet	Constructs a set from a string expression.
Subset	Returns a subset of elements from a set.
Tail	Returns a subset from the end of a set.
ToggleDrillState	Toggles the drill state of members. This function is a combination of **DrillupMember** and **DrilldownMember**.
TopCount	Returns a specified number of items from the top of a set, optionally ordering the set first.
TopPercent	Sorts a set and returns the top *n* elements whose cumulative total is at least a specified percentage.
TopSum	Sorts a set and returns the top *n* elements whose cumulative total is at least a specified value.
Union	Returns the union of two sets, optionally retaining duplicates.
VisualTotals	Dynamically totals child members specified in a set using a pattern for the total label in the result set.

Wtd	A shortcut function for the **PeriodsToDate** function that specifies the level to be Week.
Ytd	A shortcut function for the **PeriodsToDate** function that specifies the level to be Year.

String Functions

Function	Description
CalculationPassValue	Returns the value of an MDX expression evaluated over the specified calculation pass of a cube.
CoalesceEmpty	Coalesces an empty cell value to a string or number.
Generate	Returns a concatenated string created by evaluating a string expression over a set.
IIf	Returns one of two string or numeric values determined by a logical test.
LookupCube	Returns the value of an MDX expression evaluated over another specified cube in the same database.
MemberToStr	Constructs a string from a member.
Name	Returns the name of a dimension, hierarchy, level, or member.
Properties	Returns a string containing a member property value.
SetToStr	Constructs a string from a set.
TupleToStr	Constructs a string from a tuple.
UniqueName	Returns the unique name of a dimension, level, or member.
UserName	Returns the domain name and user name of the current connection.

Tuple Functions

Function	Description
Current	Returns the current tuple from a set during an iteration.

Item	Returns a tuple from a set.
StrToTuple	Constructs a tuple from a string.

Analysis Services

A

AddCalculatedMembers

Returns a set generated by adding calculated members to a specified set.

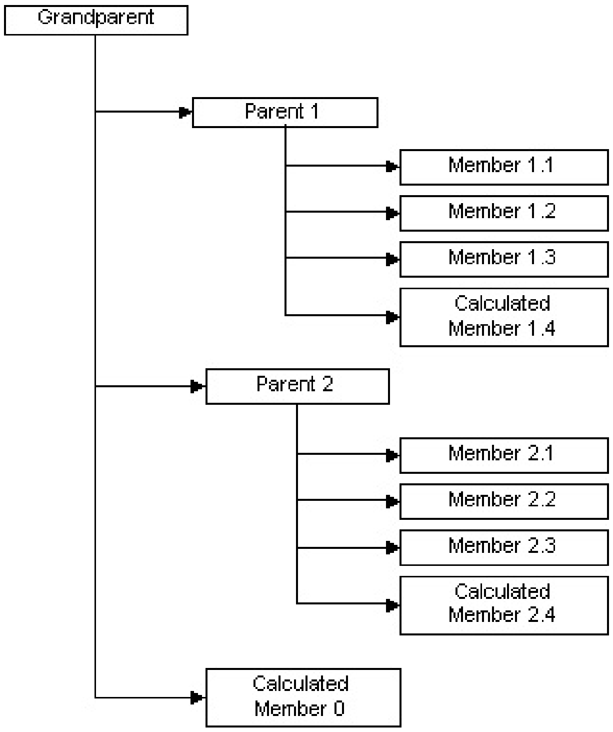
Syntax

AddCalculatedMembers(«Set»)

Remarks

This function includes only the calculated members that are siblings in the set.

Example



In the preceding diagram, the following functions return the following values.

Expression	Returns
AddCalculatedMembers({GrandParent.Children})	{[Parent 1],[Pare

AddCalculatedMembers({[Parent 1].Children})	{[Member 1.1], [
AddCalculatedMembers({[Parent 2].[Member 2.1]})	{[Member 2.1], [

Aggregate

Returns a calculated value using the appropriate aggregate function, based on the aggregation type of the member.

Syntax

Aggregate(«Set»[, «Numeric Expression»])

Remarks

This function cannot be used on calculated members.

Example

In the following expression, the calculated member Total is displayed first against the measure SumSales and then against the measure MaxSales. In the former case, Total is calculated by adding (with **Sum**). In the latter case, Total is calculated by taking the maximum.

```
WITH MEMBER Geography.Total AS  'AGGREGATE({USA, France
SELECT {Measures.SumSales, Measures.MaxSales} ON COLUMNS
    {USA, France, Total} ON ROWS
FROM SalesCube
WHERE ([1998])
```

Analysis Services

AllMembers

Returns a set containing all members of the specified dimension or level, including calculated members.

Syntax

Dimension

«Dimension».**AllMembers**

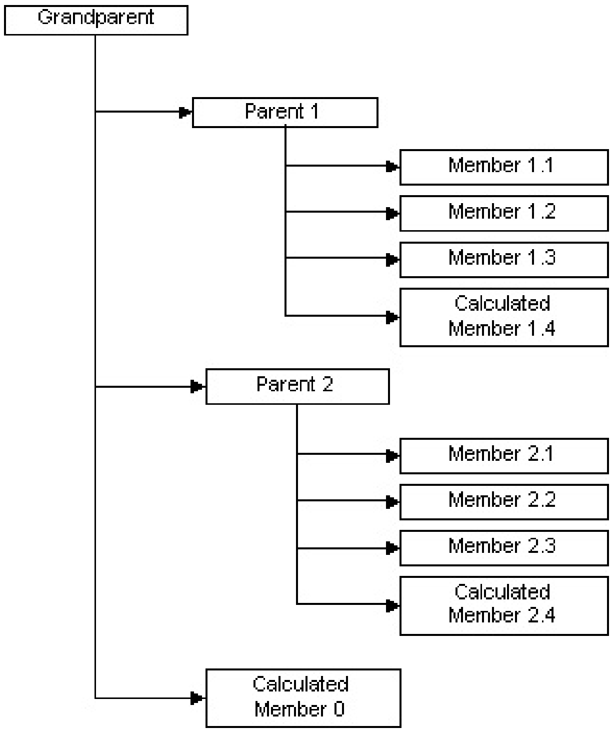
Level

«Level».**AllMembers**

Remarks

Semantically similar to the **AddCalculatedMembers** function, this function also includes only the calculated members that are siblings in the dimension or level. However, if there are no members, such as with the [Measures] dimension, the **AllMembers** function will still return the calculated members.

Example



In this diagram, the following example returns the set {[Parent 1],[Parent 2], [Calculated Member 0]}:

[GrandParent].AllMembers

Ancestor

Returns the ancestor of a member at a specified level or distance.

Syntax

Level

Ancestor(«Member», «Level»)

Returns the ancestor of «Member» from the dimension level specified in «Level».

Distance

Ancestor(«Member», «Numeric Expression»)

Returns the ancestor of «Member» that is «Numeric Expression» steps away in the hierarchy.

Example

If the Geography dimension includes levels Country, State, and City, the following functions return the following values.

Expression	**Returns**
Ancestor(Los Angeles, Country)	[USA]
Ancestor(Los Angeles, State)	[California]
Ancestor(Los Angeles, 0)	[Los Angeles]
Ancestor(Los Angeles, 1)	[California]
Ancestor(Los Angeles, 2)	[USA]

Ancestors

Returns a set of all the ancestors of a member at a specified level or distance.

Syntax

Level

Ancestors(«Member», «Level»)

Returns all ancestors of «Member» at the level specified in «Level».

The set of returned members must all be from the same hierarchy, but «Level» does not need to be a level of the same hierarchy as «Member».

Distance

Ancestors(«Member», «Numeric Expression»)

Returns all members of the hierarchy that are «Numeric Expression» steps above «Member» in the hierarchy. This form of the **Ancestors** function is intended for cases in which the level of the parent is unknown or cannot be named. The set of returned members must all be from the same hierarchy.

Note **Ancestors**(«Member», 0) returns «Member».

Remarks

Unlike the **Ancestor** function, **Ancestors** is a set value expression; it returns a set, not a member.

Examples

If the Geography dimension includes levels Country, State, and City, the following functions return the following values.

Expression	Returns

Ancestors([Los Angeles], Country)	{ USA }
Ancestors([Los Angeles], State)	{ California }
Ancestors([Los Angeles], 0)	{ [Los Angeles] }
Ancestors([Los Angeles], 1)	{ California }
Ancestors([Los Angeles], 2)	{ USA }

Ascendants

Returns the set of the ascendants of a specified member.

Syntax

Ascendants(«Member»)

Remarks

Unlike the **Ancestor** function, which returns a specific ascendant member, or ancestor, at a specific level, the **Ascendants** function performs a post-order traversal of the member hierarchy, returning all of the ascendant members related to the given member, including itself, in a set.

Example

The following example returns the set { LA, California, USA, [All Locations] }:

```
Ascendants([Los Angeles])
```

Avg

Returns the average value of a numeric expression evaluated over a set.

Syntax

Avg(«Set»[, «Numeric Expression»])

Remarks

This function requires an implicit count of the number of cells, not including empty cells. In order to force the inclusion of empty cells, the application must use the **CoalesceEmpty** function. For more information about empty cells, see the OLE DB documentation.

Example

The following function returns 2000 if respective Sales are 1000, 2000, and 3000:

```
Avg({USA, Canada, Mexico}, Sales)
```

Axis

Returns a set defined in an axis.

Syntax

Axis(«Numeric Expression»)

Remarks

The **Axis** function uses the zero-based position of an axis, specified in «Numeric Expression», to return the set defined in the axis. For example, Axis(0) returns the COLUMNS axis, Axis(1) returns the ROWS axis, and so on. This function cannot be used on the filter axis.

Example

The examples in the table use the following Multidimensional Expressions (MDX) query.

```
SELECT {Time.Members} ON COLUMNS, {Location.Members} ON
FROM TestCube
```

Expression	**Returns**
Axis(0)	The set of members represented by {Time.Members}, in the COLUMNS axis.
Axis(1)	The set of members represented by {Location.Members}, in the ROWS axis.

Analysis Services

B

BottomCount

Returns a specified number of items from the bottom of a set, optionally ordering the set first.

Syntax

BottomCount(«Set», «Count»[, «Numeric Expression»])

Remarks

This function sorts a set according to the value of «Numeric Expression» and returns the bottom «Count» members, where «Count» is a numeric expression.

IMPORTANT The **BottomCount** function, as with the **TopCount** function, always breaks the hierarchy.

Example

BottomCount(Geography.Cities.Members, 5, Sales)

See Also

TopCount

BottomPercent

Sorts a set and returns the specified number of bottommost elements whose cumulative total is at least a specified percentage.

Syntax

BottomPercent(«Set», «Percentage», «Numeric Expression»)

Remarks

This function sorts a set specified in «Set» and returns the specified number of bottommost elements whose cumulative total of «Numeric Expression» is at least «Percentage». «Percentage» is a numeric expression.

Important The **BottomPercent** function, as with the **TopPercent** function, always breaks the hierarchy.

Example

```
BottomPercent(Products.[Product Description].Members, 25, Sales)
```

See Also

TopPercent

BottomSum

Sorts a set using a numeric expression and returns the specified number of bottommost elements whose sum is at least a specified value.

Syntax

BottomSum(«Set», «Value», «Numeric Expression»)

Remarks

This function sorts on «Numeric Expression» and picks up the specified number of bottommost (the smallest number possible) elements such that their sum is at least «Value».

Important The **BottomSum** function, as with the **TopSum** function, always breaks the hierarchy.

Example

```
BottomSum(Products.[Product Description].Members, 100000, Quanti
```

See Also

TopSum

Analysis Services

C

CalculationCurrentPass

Returns the current calculation pass of a cube for the specified query context.

Syntax

CalculationCurrentPass()

Remarks

The **CalculationCurrentPass** function returns the zero-based index of the calculation pass of the current query context flag.

Example

The following example returns the current calculation pass number:

```
CalculationCurrentPass()
```

See Also

CalculationPassValue

CalculationPassValue

Returns the value of a Multidimensional Expressions (MDX) expression evaluated over the specified calculation pass of a cube.

Syntax

Numeric

CalculationPassValue(«Numeric Expression», «Pass Value»[, «Access Flag»])

The **CalculationPassValue** function returns a numeric value, evaluating the MDX numeric expression specified in «Numeric Expression» in the calculation pass specified in «Pass Value», optionally modified by an access flag specified in «Access Flags».

String

CalculationPassValue(«String Expression», «Pass Value»[, «Access Flag»])

The **CalculationPassValue** function returns a string value, evaluating the MDX string expression specified in «Numeric Expression» in the calculation pass specified in «Pass Value», optionally modified by an access flag specified in «Access Flags».

Remarks

The behavior of the «Pass Value» parameter can change depending on the content of the «Access Flag» parameter. The following table details the access flags allowed in the «Access Flag» parameter.

Access flag	Description
ABSOLUTE	The «Pass Value» contains the zero-based index of a calculation pass.
RELATIVE	The «Pass Value» contains a relative offset from the current calculation

	pass. If the offset resolves into a calculation pass index less than 0, calculation pass 0 is used; no error occurs.

Example

Numeric

The following example returns the value of [Sales] from calculation pass 0:

CalculationPassValue([Sales], 0)

String

The following example returns the full name of the [Sales] member from the previous calculation pass 0:

CalculationPassValue(MemberToStr([Sales]), -1, RELATIVE)

See Also

CalculationCurrentPass

Call

Executes a void-returning user-defined function.

Syntax

Call «UDF Name»

Remarks

The **Call** function executes the registered user-defined function specified by «UDF». This function is designed for use only with void-returning user-defined functions.

Note If the user-defined function is not registered on the client, the **Call** function will attempt to call the user-defined function from the Analysis server.

Example

```
Call MyOwnVoidFunction()
```

Children

Returns the children of a member.

Syntax

«Member».**Children**

Example

The following code returns { Nebraska, Oklahoma, Montana }:

```
[Geography].[All Geography].[Central Region].Children
```

ClosingPeriod

Returns the last sibling among the descendants of a member at a specified level.

Syntax

ClosingPeriod([«Level»[, «Member»]])

Remarks

The dimension that contains «Level» is used if «Level» is specified; otherwise, the Time dimension is used. If no «Level» is specified, the level below that of «Member» is used. If no «Level» or «Member» is specified, the default is Time.**CurrentMember**.

This function is equivalent to **BottomCount(Descendants(**«Member»**,** «Level»**), 1)**.

The **OpeningPeriod** function is similar, except that it returns the first sibling instead of the last sibling.

Example

The following example returns [1991].December:

```
ClosingPeriod(Month, [1991])
```

See Also

OpeningPeriod

CoalesceEmpty

Coalesces an empty cell value to a number or string and returns the coalesced value.

Syntax

Numeric

CoalesceEmpty(«Numeric Expression»[, «Numeric Expression»]...)

String

CoalesceEmpty(«String Expression»[, «String Expression»]...)

Remarks

This function returns the first (from the left) nonempty value expression in the list of value expressions. Returns the empty cell value if all value expressions evaluate to the empty cell value.

All value expressions must evaluate to a numeric data type or to the empty cell value. Alternatively, all value expressions must evaluate to a string data type or to the empty cell value.

For more information about empty cells, see the OLE DB documentation.

Examples

Number

The following example returns -99 if Measures.CurrentMember is the empty cell value, otherwise it returns Measures.CurrentMember :

```
CoalesceEmpty(Measures.CurrentMember, -99)
```

String

The following example returns the string "EMPTY" if Time.Parent.Name is the empty cell value, Time.Parent.Name otherwise:

```
CoalesceEmpty(Time.Parent.Name, "EMPTY")
```

Correlation

Returns the correlation of two series evaluated over a set.

Syntax

Correlation(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

This function evaluates «Set» against the first «Numeric Expression» to get the values for the y-axis. «Set» is evaluated against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the members of «Set» are used as values for the x-axis.

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

Count

Returns the number of items in a collection, depending on the collection.

Syntax

Dimension

Dimensions.**Count**

Returns the number of dimensions in a cube, including the [Measures] dimension.

Level

«Dimension»|«Hierarchy».Levels.**Count**

Returns the number of levels in a dimension or hierarchy, including the [All] level if applicable.

Set - Syntax 1

Count(«Set»[, **ExcludeEmpty** | **IncludeEmpty**])

Returns the number of cells in a set. This syntax allows empty cells to be excluded or included with the use of the **ExcludeEmpty** or **IncludeEmpty** flags, respectively.

Set - Syntax 2

«Set»**.Count**

Returns the number of cells in a set, with empty cells included.

Tuple

«Tuple».**Count**

Returns the number of dimensions in a tuple.

Note Empty cells are counted by default. The comparable **Count** function in OLE DB excludes empty cells by default.

Remarks

To exclude empty cells in the count of a set, use the optional **ExcludeEmpty** keyword.

Example

If Time has levels Year and Month, and the members of Year are 1994 and 1995, the following examples return 24:

Set - Example 1

Count({Time.Month.Members})

Set - Example 2

Time.Month.Members.Count

Cousin

Returns the child member with the same relative position under a parent member as the specified child member.

Syntax

Cousin(«Member1», «Member2»)

Remarks

This function operates on the order and position of members within levels. If two dimensions exist, in which the first one has four levels and the second one has five levels, the cousin of the third level of the first dimension is the third level of the second dimension.

Example

The following example assumes that both years 1996 and 1994 contain the same number of months preceding the member March:

```
Cousin([1996].March, [1994])
```

This example yields the member [1994].March.

If the same example assumes that the 1996 level contains the January, February, March, April, May, June, July, August, September, October, November, and December members, and the 1994 level contains the [1st Quarter], [2nd Quarter], [3rd Quarter], and [4th Quarter] members, the example returns [1994].[3rd Quarter] because it is in the same relative position (third) within the level.

Covariance

Returns the population covariance of two series evaluated over a set, using the biased population formula.

Syntax

Covariance(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

This function evaluates «Set» against the first «Numeric Expression» to get the values for the y-axis. «Set» is evaluated against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the members of «Set» are used as values for the x-axis.

Covariance uses the biased population formula. **CovarianceN** uses the unbiased population formula.

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

CovarianceN

Returns the sample covariance of two series evaluated over a set, using the unbiased population formula.

Syntax

CovarianceN(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

This function evaluates «Set» against the first «Numeric Expression» to get the values for the y-axis. «Set» is evaluated against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the members of «Set» are used as values for the x-axis.

The **CovarianceN** function uses the unbiased population formula, while the **Covariance** function uses the biased population formula.

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

Crossjoin

Returns the cross product of two sets.

Syntax

Crossjoin(«Set1», «Set2»)

Alternate Syntax

«Set1» * «Set2»

Note This alternate syntax is a Microsoft® SQL Server™ 2000 Analysis Services extension to Multidimensional Expressions (MDX) in OLE DB 2.0 and later.

Remarks

The order of tuples in the resulting set depends on the order of «Set1» and «Set2» and the order of their members.

If «Set1» = {*x1*, *x2*,...,*xn*} and «Set2» = {*y1*, *y2*, ..., *yn*}, then **Crossjoin**(*Set1*, *Set2*) is:

{(*x1*, *y1*), (*x1*, *y2*),...,(*x1*, *yn*), (*x2*, *y1*), (*x2*, *y2*),...,

(*x2*, *yn*),..., (*xn*, *y1*), (*xn*, *y2*),..., (*xn*, *yn*)}

Example

The following example returns {([1994], USA), ([1994], Japan), ([1995], USA), ([1995], Japan)}:

CrossJoin({[1994], [1995]}, {USA, Japan})

Current

Returns the current tuple from a set during an iteration.

Syntax

«Set».**Current**

Remarks

During an iteration through a set, as in functions such as **Generate**, at each step in the iteration the tuple being operated upon is the current tuple. This function returns that tuple.

CurrentMember

Returns the current member along a dimension during an iteration.

Syntax

«Dimension».**CurrentMember**

Remarks

During iteration through a set of dimension members, at each step in the iteration, the member being operated upon is the current member. This function returns that member.

Example

```
Time.[1996].CurrentMember
```

D

DataMember

Returns the system-generated data member that is associated with a nonleaf member of a dimension.

Syntax

«Member».**DataMember**

Remarks

Returns «Member» if «Member» is a leaf member, or if the nonleaf member does not have an associated data member.

Example

The following example returns the data member for the [Location].[UK] nonleaf member:

```
[Location].[UK].DataMember
```

DefaultMember

Returns the default member of a dimension or hierarchy.

Syntax

Dimension

«Dimension».**DefaultMember**

Hierarchy

«Hierarchy».**DefaultMember**

Remarks

The default member is specified in the **Default Member** property of the dimension or hierarchy (**DefaultMember** in the Decision Support Objects library.) If this property is empty and the dimension contains an (All) level, the default member is the All member. If this property is empty and the dimension or hierarchy does not contain an (All) level, the default member is an arbitrary member of the highest level. In this last case, the **DefaultMember** function is ambiguous.

Example

Dimension

If the Time dimension has levels (All), Year, Quarter, and Month, and member All-Time, the following expression returns All-Time:

```
Time.DefaultMember
```

Hierarchy

If the [Fiscal Year] hierarchy has levels Quarter, and Month, and member

[Month 1], the following expression returns [Month 1]:

```
[Fiscal Year].DefaultMember
```

Descendants

Returns the set of descendants of a member at a specified level or distance, optionally including or excluding descendants in other levels.

Syntax

Level

Descendants(«Member», [«Level»[, «Desc_flags»]])

Returns the set of descendants of a member, specified by «Member», at the level specified by «Level», optionally modified by a flag specified in «Desc_flags».

If no «Level» or «Desc-flags» arguments are specified, the function behaves as in following syntax:

Descendants(«Member», «Member».Level, SELF_BEFORE_AND_AFTER)

Distance

Descendants(«Member», «Distance»[, «Desc_flags»])

Returns the descendants of a member, specified by «Member», that are «Distance» steps away in the hierarchy, optionally modified by a flag specified in «Desc_flags». This syntax is typically used to deal with ragged hierarchies. Specifying a «Distance» of 0 returns a set consisting only of the member specified in «Member».

Flags

Flag	**Description**
SELF	Default. Returns descendant members from «Level» only. Includes «Member», if and only if «Level» specified is the level of «Member».
AFTER	Returns descendant members from all levels

	subordinate to «Level».
BEFORE	Returns descendant members from all levels between «Member» and «Level», not including members from «Level».
BEFORE_AND_AFTER	Returns descendant members from all levels subordinate to the level of «Member» except members from «Level».
SELF_AND_AFTER	Returns descendant members from «Level» and all levels subordinate to «Level».
SELF_AND_BEFORE	Returns descendant members from «Level» and all levels between «Member» and «Level».
SELF_BEFORE_AFTER	Returns descendant members from all levels subordinate to the level of «Member».
LEAVES	Returns leaf descendant members between «Member» and «Level» or «Distance».

Remarks

By default, only members at the specified level or distance will be included. This function corresponds to a «desc_flags» value of SELF. By changing the value of «desc_flags», you can include or exclude descendants at the specified level or distance, the children before or the children after the specified level or distance (until the leaf node), as well as all of the leaf children regardless of the specified level or distance.

Example

Assume the levels in the Location dimension are named (in hierarchical order) Countries, States, Counties, and Cities.

Expression	**Returns**
Descendants(USA)	All states, counties and cities in USA
Descendants(USA, Counties)	All

	counties in USA
Descendants(USA, Counties, SELF)	All counties in USA
Descendants(USA, Counties, BEFORE)	All states in USA
Descendants(USA, Counties, AFTER)	All cities in USA
Descendants(USA, Counties, BEFORE_AND_AFTER)	All states and cities in USA
Descendants(USA, Counties, SELF_BEFORE_AFTER)	All states, counties, and cities in USA
Descendants(USA, States, LEAVES)	All states in USA and any leaf members between the Countries level and the States level.
Descendants(USA, 1)	All states in USA
Descendants(USA, 2, SELF_BEFORE_AFTER)	All states, counties and cities in USA

See Also

Level

Dimension

Returns the dimension that contains a specified member, level, or hierarchy.

Syntax

Member

«Member».**Dimension**

Returns the dimension that contains «Member».

Level

«Level».**Dimension**

Returns the dimension that contains «Level».

Hierarchy

«Hierarchy».**Dimension**

Returns the dimension that contains «Hierarchy».

Remarks

Microsoft® SQL Server™ 2000 Analysis Services implements hierarchies as separate dimensions, so «Hierarchy».Dimension returns «Hierarchy».

Example

Member

[1998].Dimension

This example returns Time.

Level

Year.Dimension

This example returns Time.

Hierarchy

FiscalYear.Dimension

This example returns Time.

Dimensions

Returns the dimension specified by a numeric or string expression.

Syntax

Numeric

Dimensions(«Numeric Expression»)

Returns the dimension whose zero-based position within the cube is specified by «Numeric Expression».

Note The Measures dimension is always represented by Dimensions(0).

String

Dimensions(«String Expression»)

Returns the dimension whose name is specified by «String Expression».

Remarks

The string version of the **Dimensions** function is typically used with user-defined functions.

Example

If dimensions Time, Region, and Product are added to a cube (in the listed order), the following expression returns Region:

```
Dimensions(2)
```

Distinct

Returns a set, removing duplicate tuples from a specified set.

Syntax

Distinct(«Set»)

Remarks

Duplicates are eliminated from the tail.

Example

The following example returns {(a,b), (c,d)}:

Distinct({(a,b), (c,d), (a,b)})

DistinctCount

Returns the number of distinct, non-empty tuples in a set.

Syntax

DistinctCount(«Set»)

Remarks

The **DistinctCount** function is equivalent to **Count**(**Distinct**(«Set»), ExcludeEmpty). This function can be applied only on calculated measures, and can involve only the topmost level. For example, the following MDX query will fail, because the DistinctCount function is being applied to a non-measure member:

```
WITH MEMBER Gender.a as 'DistinctCount({[Product Name].Memb
SELECT
   { Gender.a } ON COLUMNS,
   { Customers.Children } ON ROWS
FROM Sales
WHERE (Measures.[unit sales])
```

By using a calculated measure to replace the non-measure member in the previous example, the following MDX query example will work:

```
WITH MEMBER Measures.a as 'DistinctCount({[Product Name].Mem
SELECT
   { Measures.a } ON COLUMNS,
   { Customers.Children } ON ROWS
FROM Sales
WHERE (Measures.[unit sales])
```

Example

The following example returns 2, assuming all three tuples resolve to non-empty cells:

```
DistinctCount({Time.[1995], Time.[1997], Time.[1995]})
```

DrilldownLevel

Drills down the members of a set to one level below the lowest level represented in the set, or to one level below an optionally specified level of a member represented in the set.

Syntax

DrilldownLevel(«Set»[, {«Level» | , «Index»}])

Returns the hierarchized members of a set, specified in «Set», one level below the lowest level represented in the set, or to one level below an optional level, either specified by reference in «Level» or by its zero-based index in «Index», of a member represented in the set.

Remarks

The members that result from the drill down are hierarchized to occur under their parents. Order is preserved among the original members in «Set».

If «Level» is specified and there is no member of «Level» represented in «Set», «Set» is returned.

When drilling down in sets of tuples, you can specify the dimension to be drilled down by its zero-based numeric position in the tuples instead of by level. For example, in the following set, a sample tuple is (Canada, Drink, [1998]):

```
(Crossjoin(Crossjoin([Store Country].members,[Product Family].mem
[Year].members)
```

The Store dimension is position 0, the Product dimension is position 1, and the Time dimension is position 2. The following expression drills down on the Product dimension:

```
DrilldownLevel(Crossjoin(Crossjoin([Store Country].members,
[Product Family].members),[Year].members),,1)
```

Example

The following example returns the set {USA, CA, <all cities in CA>, WA, <all cities in WA>, Canada}:

DrilldownLevel({[Customers Location].USA, [Customers Location].C
[Customers Location].WA, [Customers Location].Canada})

This example returns the same set as the previous example:

DrilldownLevel({[Customers Location].USA, [Customers Location].C
[Customers Location].WA, [Customers Location].Canada},
[Customers Location].[State Province])

The following example returns the set {USA, CA, WA, Canada, BC}, assuming that the [Country] level of the [Customers Location] dimension has an index of 1:

DrilldownLevel({[Customers Location].USA, [Customers Location].C
[Customers Location].WA, [Customers Location].Canada}, ,1)

DrilldownLevelBottom

Drills down the bottommost members of a set, at a specified level, to one level below.

Syntax

DrilldownLevelBottom(«Set», «Count»[, [«Level»][, «Numeric Expression»]])

Remarks

This function is similar to the **DrilldownLevel** function, but instead of including all children for each member at the specified «Level», only the bottom «Count» of children is returned, based on «Numeric Expression».

Example

Assuming Oregon and Washington had the lowest sales, the following example returns the set {USA, Oregon, Washington}:

```
DrilldownLevelBottom({[Customers Location].USA},2,,[Unit Sales])
```

The expression

```
DrilldownLevelBottom({[Customers Location].[Country].Members},2
```

returns the set

{Canada, *<bottom two provinces in Canada>*,
Mexico, *<bottom two states in Mexico>*, ...,
USA, *<bottom two states in USA>*}.

See Also

DrilldownLevel

DrilldownLevelTop

Drills down the topmost members of a set, at a specified level, to one level below.

Syntax

DrilldownLevelTop(«Set», «Count»[, [«Level»][, «Numeric Expression»]])

Remarks

This function is similar to the **DrilldownLevel** function, but instead of including all children for each member at the specified «Level», only the top «Count» of children is returned, based on «Numeric Expression».

Example

Assuming California and New York had the highest sales, the following example returns the set {USA, California, New York}:

```
DrilldownLevelTop({[Customers Location].USA},2,,[Unit Sales])
```

The expression

```
DrilldownLevelTop({[Customers Location].[Country].Members},2,,[U
```

returns the set

{Canada, *<top two provinces in Canada>*,
Mexico, *< top two states in Mexico>*, ...,
USA, *< top two states in USA>*}.

See Also

DrilldownLevel

DrilldownMember

Drills down the members in a specified set that are present in a second specified set.

Alternatively, drills down on a set of tuples.

Syntax

DrilldownMember(«Set1», «Set2»[, RECURSIVE])

Remarks

This function drills down the members in «Set1» that are present in «Set2». «Set1» is usually a subset of «Set2». If RECURSIVE is specified, the drilldown continues, comparing the expanded result set against «Set2» at each step.

«Set1» may contain tuples instead of members. Tuple drilldown is an extension of OLE DB, and it returns a set of tuples instead of members.

Example

These examples drill down into members.

This example

```
DrilldownMember({USA, Canada, Mexico}, {USA, Washington, Mex
```

returns the set:

{USA, *<all states in USA>*, Canada, Mexico, *<all states in Mexico>*}

and this example

```
DrilldownMember({USA, Canada, Mexico}, {USA, Washington, Mex
```

returns the set:

{USA, *<all states in USA before Washington>*,
WA, *<all cities in Washington>*, *<all cities in USA after Washington>*
Canada, Mexico, *<all states in Mexico>*}.

The following examples drill down into tuples.

This example

DrilldownMember({(USA,[Unit Sales]), (Canada,[Unit Sales]),
(Mexico,[Unit Sales])},{USA, Washington})

returns the set of tuples:

{(USA,[Unit Sales]), (Arizona,[Unit Sales]), ... , (Wyoming,[Unit Sale
(Canada,[Unit Sales]), (Mexico,[Unit Sales])}.

and this example

DrilldownMember({(USA,[Unit Sales]), (Canada,[Unit Sales]),
(Mexico,[Unit Sales])},{USA, Washington},RECURSIVE)

returns the set of tuples:

{(USA,[Unit Sales]), (Arizona,[Unit Sales]), ... (Washington,[Unit Sal
(Wyoming,[Unit Sales]), (Canada,[Unit Sales]), (Mexico,[Unit Sales])

DrilldownMemberBottom

Drills down the members in a specified set that are present in a second specified set, limiting the result set to a specified number of members. Alternatively, it also drills down on a set of tuples.

Syntax

DrilldownMemberBottom(«Set1», «Set2», «Count»[, [«Numeric Expression»] [, RECURSIVE]])

Remarks

«Set1» can contain tuples instead of members. Tuple drilldown is an extension of OLE DB, and it returns a set of tuples instead of members.

This function is similar to the **DrilldownMember** function, but instead of including all children of a member, only the bottom «Count» of children is returned, based on «Numeric Expression».

Example

This example

```
DrilldownMemberBottom({USA, Canada, Mexico}, {USA, Washingt
```

returns the set

{USA, *<bottom two states in USA>*, Canada, Mexico, *<bottom two st*

and this example

```
DrilldownMemberBottom({USA, Washington, Canada, Mexico}, {US
RECURSIVE)
```

returns the set

{USA, Washington, *<bottom two cities in WA>*, Canada, Mexico, *<bo*

Tuple drilldown

This example

DrilldownMemberBottom({(USA, [Unit Sales]), (Washington, [Unit S
(Canada, [Unit Sales]), (Mexico, [Unit Sales])},
{USA, Washington},2,[Store Sales],RECURSIVE)

returns the set

{(USA, [Unit Sales]), (Washington, [Unit Sales]), (*<bottom two cities*
[Unit Sales]), (Canada, [Unit Sales]), (Mexico, [Unit Sales]}.

See Also

DrilldownMember

DrilldownMemberTop

Drills down the members in a specified set that are present in a second specified set, limiting the result set to a specified number of members.

Alternatively, drills down on a set of tuples.

Syntax

DrilldownMemberTop(«Set1», «Set2», «Count»[, [«Numeric Expression»][, RECURSIVE]])

Remarks

«Set1» can contain tuples instead of members. Tuple drilldown is an extension of OLE DB, and it returns a set of tuples instead of members.

This function is similar to the **DrilldownMember** function, but instead of including all children of a member, only the top «Count» of children is returned, based on «Numeric Expression».

Example

Member Drilldown

The following examples demonstrate member drilldown:

Expression
DrilldownMemberTop({USA, Canada, Mexico}, {USA, Washington,
DrilldownMemberTop({USA, Washington, Canada, Mexico}, {USA,

Tuple drilldown

This example

DrilldownMemberTop({(USA, [Unit Sales]), (Washington, [Unit Sales
(Canada, [Unit Sales]), (Mexico, [Unit Sales])},
{USA, Washington},2,[Store Sales],RECURSIVE)

returns the set

{(USA, [Unit Sales]), (Washington, [Unit Sales]), (<*top two cities in W*
[Unit Sales]), (Canada, [Unit Sales]), (Mexico, [Unit Sales]}.

See Also

DrilldownMember

DrillupLevel

Drills up the members of a set that are below a specified level.

Syntax

DrillupLevel(«Set»[, «Level»])

Remarks

If «Level» is not specified, it is assumed to be the level immediately above the level of the lowest level member in «Set».

Example

This example

```
DrillUpLevel({USA, California, [Los Angeles], Washington, Seattle, C
```

returns the set

```
{USA, California, Washington, Canada, [British Columbia]}
```

and this example

```
DrillUpLevel({USA, California, [Los Angeles], Washington, Seattle, C
```

returns the set

```
{USA, Canada }.
```

DrillupMember

Drills up the members in a specified set that are present in a second specified set.

Syntax

DrillupMember(«Set1», «Set2»)

Remarks

This function drills up the members in «Set1» that are present in «Set2». «Set2» is usually a subset of «Set1».

Example

This example

```
DrillupMember({Canada, Mexico, USA, Washington, Seattle},{Washi
```

returns the set

```
{Canada, Mexico, USA, Washington}.
```

E

Except

Finds the difference between two sets, optionally retaining duplicates.

Syntax

Except(«Set1», «Set2»[, ALL])

Remarks

Duplicates are eliminated from both sets prior to finding the difference. The optional ALL flag retains duplicates. Matching duplicates in «Set1» are eliminated and nonmatching duplicates are retained.

Example

This example

Except({Canada, [British Columbia], Mexico, [British Columbia], US

returns

{[British Columbia], USA, Washington}

and this example

Except({Canada, [British Columbia], Mexico, [British Columbia], US

returns

{[British Columbia], [British Columbia], USA, Washington}.

Extract

Returns a set of tuples from extracted dimension elements.

Syntax

Extract(«Set», «Dimension»[, «Dimension»...])

Remarks

This function returns a set consisting of tuples from the extracted «Dimension» elements. This function always removes the duplicates. The **Extract** function performs the opposite action of the **Crossjoin** function.

Example

This example

```
Extract({([1997], Washington), ([1997], California), ([1998], Californi
```

returns the set

```
{[1997], [1998]}
```

See Also

Crossjoin

F

Filter

Returns the set resulting from filtering a specified set based on a search condition.

Syntax

Filter(«Set», «Search Condition»)

Remarks

The **Filter** function evaluates a Multidimensional Expressions (MDX) logical expression, specified in «Search Condition», against each member of the set specified in «Set», returning the set of members that met the search condition.

The **Filter** function works in a fashion similar to that of the **IIf** function. While the **IIf** function returns only one of two options based on the evaluation of an MDX logical expression, the **Filter** function returns a set of members that meet the specified search condition. In effect, the **Filter** function executes **IIf**(«Search Criteria», «Member», NULL) on each member in the set and returns the resulting set. If none of the members meet the search condition, an empty set is returned.

Example

The following example returns {Paris, Buffalo} if these cities declined in sales from the 1995 to the 1996 level:

```
Filter(SampleSet, (Sales,[1996]) < (Sales, [1995]))
```

See Also

IIf

FirstChild

Returns the first child of a member.

Syntax

«Member».**FirstChild**

Example

If the Time dimension includes the levels Year, Quarter, Month, Week, and Day, the following code returns January:

```
[1995].FirstChild
```

FirstSibling

Returns the first child of the parent of a member.

Syntax

«Member».**FirstSibling**

Example

Assuming a dimension composed of months, the following example returns January:

May.FirstSibling

G

Generate

Applies a set to each member of another set and joins the resulting sets by union. Alternatively, returns a concatenated string created by evaluating a string expression over a set.

Syntax

Set

Generate(«Set1», «Set2»[, ALL])

String

Generate(«Set», «String Expression»[, «Delimiter»])

Remarks

The set version of this function applies «Set2» to each member of «Set1» and joins the resulting sets by union. If ALL is specified, duplicates in the result are retained.

The string version of this function iterates through each member of the set specified in «Set», evaluates a string expression, specified in «String Expression», against the member and concatenates the result into the return string. Optionally, the string can be delimited by supplying a string expression in «Delimiter», separating each result in the concatenated return string.

Examples

Set

```
Generate({USA, France}, Descendants(Geography.CurrentMember, C
```

For each member of the set {USA, France}, this function applies the expression **Descendants**(Geography.**CurrentMember**, Cities). Each such application results in a set. (Application to USA will generate the set of all cities in USA; application to France will generate all cities in France.) These sets are joined by union to return the result of this function. In this example, all cities in USA and France will be the result. In general, **Generate**(«Set1», «set_expression») will apply «set_expression» to each member of «Set1» and join the results by union.

If «Set1» is not related to «set_expression» by means of **CurrentMember**, then **Generate** results in a simple replication of the set implied by «set_expression», with as many replications as there are tuples in «Set1». If the optional ALL flag is specified, all duplicates are retained in the result. If ALL is not specified, duplicates are removed. For example,

```
Generate({USA, FRANCE}, {SEATTLE, BOSTON}, ALL)
```

returns the set

```
{SEATTLE, BOSTON, SEATTLE, BOSTON}.
```

However, if ALL was not specified, then the set returned is

```
{SEATTLE, BOSTON}.
```

String

The following example returns the string "19971998":

```
Generate( {Time.[1997], Time.[1998]}, Time.CurrentMember.Name)
```

The following example returns the string "1997 and 1998":

```
Generate( {Time.[1997], Time.[1998]}, Time.CurrentMember.Name, "
```

H

Head

Returns the first specified number of elements in a set.

Syntax

Head(«Set»[, «Numeric Expression»])

Remarks

This function returns the first «Numeric Expression» elements in a set. The order of elements is preserved. The default value of «Numeric Expression» is 1. If «Numeric Expression» is less than 1, the empty set is returned. If «Numeric Expression» exceeds the number of tuples in the set, the original set is returned.

Example

The following example returns the set {USA, Canada, France}:

```
Head({USA, Canada, France, Germany, Japan}, 3)
```

Hierarchize

Orders the members of a set in a hierarchy.

Syntax

Hierarchize(«Set»[, POST])

Remarks

This function orders the members of «Set» in a hierarchy. Unless the POST keyword is used, members in a level are sorted in their natural order, which is the default ordering of the members along a dimension when no other sort conditions are specified. The POST keyword uses a post-natural order for the sorting of members in a level. **Hierarchize** will always retain duplicates.

Example

This example

Hierarchize(SampleSet)

returns the set in natural order. The hierarchized dataset follows (assuming that the natural order for the data source is alphabetical):

France		
		Nice
		Paris
UK		
		London
USA		
	California	
		LA
		Buffalo
		NYC

Hierarchy

Returns the hierarchy that contains a specified member or level.

Syntax

Member

«Member».**Hierarchy**

Returns the hierarchy that contains «Member».

Level

«Level».**Hierarchy**

Returns the hierarchy that contains «Level».

Example

Member

```
[January].Hierarchy
```

This example returns Time.FiscalYear.

Level

```
[Quarter].Hierarchy
```

This example returns Time.FiscalYear.

Analysis Services

I

Ignore

Reserved.

IIf

Returns one of two numeric or string values determined by a logical test.

Syntax

Numeric

IIf(«Logical Expression», «Numeric Expression1», «Numeric Expression2»)

This function returns «Numeric Expression1» if «Logical Expression» evaluates to TRUE, otherwise returns «Numeric Expression2».

String

IIf(«Logical Expression», «String Expression1», «String Expression2»)

This function returns «String Expression1» if «Logical Expression» evaluates to TRUE, otherwise returns «String Expression2».

Remarks

«Logical Expression» is considered to be FALSE only if its value is zero. Any other value is interpreted as TRUE.

The **Iif** function is not recommended for creating a set of members based on search criteria. Instead, use the **Filter** function to evaluate each member in a specified set against a logical expression and return a subset of members.

Examples

Numeric

The following example returns 0 if Measures.CurrentMember is an empty cell, 1 otherwise:

```
IIf(IsEmpty(Measures.CurrentMember), 0, 1)
```

String

The following string returns the string "Yes" if Measures.CurrentMember is an empty cell, the string, "No" otherwise:

```
IIf(IsEmpty(Measures.CurrentMember), "Yes", "No")
```

See Also

Filter

Intersect

Returns the intersection of two input sets, optionally retaining duplicates.

Syntax

Intersect(«Set1», «Set2»[, ALL])

Remarks

This function returns the intersection of «Set1» and «Set2». By default, duplicates are eliminated from both sets prior to intersection.

The optional ALL retains duplicates. There are several ways for ALL to work. The algorithm is: Nonduplicated elements are intersected as usual. For each duplicate in «Set1», match it with a duplicate in «Set2», if one exists, and keep matching duplicates in the intersected set.

Example

This example

```
Intersect({[1994], [1995], [1996]}, {[1995], [1996], [1997]})
```

returns the set {[1995], [1996]}.

Is

Returns TRUE if two compared objects are equivalent, FALSE otherwise.

Syntax

«Object 1» **IS** «Object 2»

Returns TRUE if the two tuples or members specified in «Object 1» and «Object 2» are equivalent, FALSE otherwise.

Alternate Syntax

«Object 1» **IS NULL**

Returns TRUE if the level, tuple, or member specified in «Object 1» is NULL, FALSE otherwise.

Remarks

The **Is** function is typically used for tuples and members to determine whether the objects are idempotent.

Example

The following example returns FALSE:

```
Time.[1996] IS NULL
```

IsAncestor

Returns TRUE if a specified member is an ancestor of another specified member, FALSE otherwise.

Syntax

IsAncestor(«Member1»,«Member2»)

Remarks

This function returns TRUE if the member indicated in «Member 1» is an ancestor of the member specified in «Member 2».

Example

The following example returns TRUE if [Time].CurrentMember is an ancestor of [Time].[January]:

```
IsAncestor([Time].CurrentMember, [Time].[January])
```

See Also

Ancestor

IsEmpty

Returns TRUE if the evaluated expression is the empty cell value, FALSE otherwise.

Syntax

IsEmpty(«Value Expression»)

Remarks

The **IsEmpty** function is the only way to reliably test for an empty cell, because the empty cell value has special meaning to Microsoft® SQL Server™ 2000 Analysis Services. For more information about empty cells, see the OLE DB documentation.

Example

The following example returns TRUE if Measures.CurrentMember is an empty cell:

```
IsEmpty(Measures.CurrentMember)
```

See Also

Working with Empty Cells

IsGeneration

Returns TRUE if a specified member is in a specified generation, FALSE otherwise.

Syntax

IsGeneration(«Member»,«Numeric Expression»)

Remarks

This function returns TRUE if the member indicated in «Member» is in the generation specified in «Numeric Expression». For the purposes of generation indexing, leaf members are generation index 0. All other members are part of the reunion of their children generation + 1. Because of this, a specific member could belong to more than one generation.

Example

The following example returns TRUE if [Time].CurrentMember is part of the second generation:

```
IsGeneration([Time].CurrentMember, 2)
```

IsLeaf

Returns TRUE if a specified member is a leaf member, FALSE otherwise.

Syntax

IsLeaf(«Member»)

Remarks

This function returns TRUE if the member indicated in «Member» is a leaf member.

Example

The following example returns TRUE if [Time].CurrentMember is a leaf member:

```
IsLeaf([Time].CurrentMember)
```

IsSibling

Returns TRUE if a specified member is a sibling of another specified member, FALSE otherwise.

Syntax

IsSibling(«Member1»,«Member2»)

Remarks

This function returns TRUE if the member indicated in «Member 1» is a sibling of the member specified in «Member 2».

Example

The following example returns TRUE if [Time].[1995] is a sibling of [Time].[1997]:

```
IsSibling([Time].[1995], [Time].[1997])
```

Item

Returns a member from a specified tuple. Alternatively, returns a tuple from a set.

Syntax

Member

«Tuple».**Item**(«Index»)

Returns a member from the tuple specified in «Tuple». The member to be returned is specified by the zero-based position of the member in the tuple in «Index».

Tuple

«Set».**Item**(«String Expression»[, «String Expression»...] | «Index»)

Returns a tuple from the set specified in «Set». The tuple to be returned is specified either by name in «String Expression», or by the zero-based position of the tuple in the set in «Index».

Examples

Member

The following example returns [1999]:

```
([1999],Sales, [2000],Sales).Item(0)
```

The following example returns [1996], if [1996] is the first member in the Year level of the Time dimension:

```
Time.Year.Members.Item(0)
```

Tuple

The following example returns ([1996],Sales):

```
{([1996],Sales), ([1997],Sales), ([1998],Sales)}.Item(0)
```

L

Lag

Returns the member that is a specified number of positions prior to a specified member along the dimension of the member.

Syntax

«Member».**Lag**(«Numeric Expression»)

Remarks

Member positions in the dimension are determined by the dimension's natural order. The numbering of the positions is zero-based.

If «Numeric Expression» is zero, «Member» is returned. If «Numeric Expression» is negative, a subsequent member is returned.

Lag(1) is equivalent to **PrevMember**. **Lag**(-1) is equivalent to **NextMember**.

The «Member»**.Lead** function is similar, except that it looks in the opposite direction. **Lag**(*n*) is equivalent to **Lead**(-*n*).

Example

If the levels in the Time dimension include Year and Month, the following example returns [1994].November:

```
[1995].February.Lag(3)
```

LastChild

Returns the last child of a specified member.

Syntax

«Member».**LastChild**

Example

If a Time dimension includes Year, Quarter, Month, Week, and Day, the following example returns December:

1995.LastChild

LastPeriods

Returns a set of members prior to and including a specified member.

Syntax

LastPeriods(«Index»[, «Member»])

Remarks

If «Member» is not specified, it is **Time.CurrentMember**.

If «Index» is positive, returns the set of «Index» members ending with «Member» and starting with the member lagging «Index» - 1 from «Member».

If «Index» is negative, returns the set of (- «Index») members starting with «Member» and ending with the member leading (- «Index» - 1) from «Member».

If «Index» is zero, the empty set is returned.

Examples

The following table lists possible uses of the **LastPeriods** function.

Expression	**Returns**
LastPeriods(2, [1997June])	{[1997May], [1997June]}
LastPeriods(-2, [1997June])	{[1997June], [1997July]}
LastPeriods(1, [1997June])	{[1997June]}
LastPeriods(-1, [1997June])	{[1997June]}
LastPeriods(0, [1997June])	Empty set

LastSibling

Returns the last child of the parent of a specified member.

Syntax

«Member».**LastSibling**

Example

If the parent level is quarters, the following example returns June:

```
May.LastSibling
```

Lead

Returns the member that is a specified number of positions following a specified member along the dimension of the member.

Syntax

«Member».**Lead**(«Numeric Expression»)

Remarks

Member positions in the dimension are determined by the dimension's natural order. The numbering of the positions is zero-based.

If «Numeric Expression» is zero, «Member» is returned. If «Numeric Expression» is negative, a prior member is returned.

Lead(1) is equivalent to **NextMember**. **Lead**(-1) is equivalent to **PrevMember**.

The «Member»**.Lag** function is similar, except that it looks in the opposite direction. **Lead**(*n*) is equivalent to **Lag**(-*n*).

Example

If the levels in the Time dimension include Year and Month, the following example returns [1995].February:

```
[1994].November.Lead(3)
```

Level

Returns the level of a member.

Syntax

«Member».**Level**

Example

If the Time dimension has the (All), Year, Quarter, Month, Week, and Day levels, the following example returns the Month level:

```
January.Level
```

This example returns the name of the Month level:

```
January.Level.Name
```

Levels

Returns the level specified by a numeric or string expression.

Syntax

Numeric

«Dimension».**Levels**(«Numeric Expression»)

Returns the level whose zero-based position is specified by «Numeric Expression».

String

Levels(«String Expression»

Returns the level whose name is specified by «String Expression».

Remarks

Use the string version of the **Levels** function for user-defined functions.

Examples

The following examples assume that the Time dimension has (All), Year, Quarter, Month, Week and Day levels.

Numeric

The following example returns the Quarter level:

```
Time.Levels(2)
```

String

The following example returns the Year level:

Levels("Year")

LinkMember

Returns the member equivalent to a specified member in a specified hierarchy.

Syntax

LinkMember(«Member», «Hierarchy»)

Example

The following example returns [Fiscal Year].[1999].[Qtr1].[Jan].[12]:

LinkMember([Calendar].[1999].[Jan].[12], [Fiscal Year])

LinRegIntercept

Calculates the linear regression of a set and returns the value of *b* in the regression line y = ax + b.

Syntax

LinRegIntercept(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

Linear regression that uses the least-squares method calculates the equation of the best-fit line for a series of points. Let the regression line be given by the following equation, where *a* is called the slope and *b* is called the intercept:

y = ax+b

This function evaluates «Set» against the first «Numeric Expression» to get the set of values for the y-axis. It then evaluates «Set» against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the function uses the members of «Set» as values for the x-axis.

The latter case is not often useful for standard dimensions (for example, SalesPerson). However, it is often used with the Time dimension.

After obtaining the set of points, **LinRegIntercept** returns the intercept of the regression line (*b* in the equation).

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

Example

The following example returns the b value for y = ax + b formula to find the trend in Sales for the sales of the last nine periods:

LinRegIntercept(LastPeriods(9), Sales) returns the value for b.

LinRegPoint

Calculates the linear regression of a set and returns the value of *y* in the regression line y = ax + b.

Syntax

LinRegPoint(«Numeric Expression», «Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

Linear regression that uses the least-squares method calculates the equation of the best-fit line for a series of points. Let the regression line be given by the following equation, where *a* is called the slope and *b* is called the intercept:

y = ax+b

LinRegPoint uses its last three arguments like the other **LinReg***xxx* functions use them: to calculate the regression line. The function evaluates the first argument and uses the resulting number as the *x* value in the regression equation (y = ax + b) to calculate the *y* value.

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

Example

This example finds the trend in Sales for the sales of the last nine periods and forecasts the next period:

LinRegPoint(10, LastPeriods(9), Sales) returns the next period.

LinRegR2

Calculates the linear regression of a set and returns R^2 (the coefficient of determination).

Syntax

LinRegR2(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

Linear regression that uses the least-squares method calculates the equation of the best-fit line for a series of points. Let the regression line be given by the following equation, where *a* is called the slope and *b* is called the intercept:

y = ax+b

This function evaluates «Set» against the first «Numeric Expression» to get the set of values for the y-axis. It then evaluates «Set» against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the function uses the members of «Set» as values for the x-axis.

The latter case is not often useful for standard dimensions (for example, SalesPerson). However, it is often used with the Time dimension.

After obtaining the set of points, **LinRegR2** returns the statistical R^2 that describes the fit of the linear equation to the points.

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

LinRegSlope

Calculates the linear regression of a set and returns the value of *a* in the regression line y = ax + b.

Syntax

LinRegSlope(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

Linear regression that uses the least-squares method calculates the equation of the best-fit line for a series of points. Let the regression line be given by the following equation, where *a* is called the slope and *b* is called the intercept:

y = ax+b

This function evaluates «Set» against the first «Numeric Expression» to get the set of values for the y-axis. It then evaluates «Set» against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the function uses the members of «Set» as values for the x-axis.

The latter case is not often useful for standard dimensions (for example, SalesPerson). However, it is often used with the Time dimension.

After obtaining the set of points, **LinRegSlope** returns the slope of the regression line (*a* in the equation).

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

Example

The following example finds the trend in Sales for the sales of the last nine periods and returns the value for a in the formula y = ax + b:

LinRegSlope(LastPeriods(9), Sales) returns the value for a.

LinRegVariance

Calculates the linear regression of a set and returns the variance associated with the regression line y = ax + b.

Syntax

LinRegVariance(«Set», «Numeric Expression»[, «Numeric Expression»])

Remarks

Linear regression that uses the least-squares method calculates the equation of the best-fit line for a series of points. Let the regression line be given by the following equation, where *a* is called the slope and *b* is called the intercept:

y = ax+b

This function evaluates «Set» against the first «Numeric Expression» to get the set of values for the y-axis. It then evaluates «Set» against the second «Numeric Expression», if present, to get the set of values for the x-axis. If the second «Numeric Expression» is not present, the function uses the members of «Set» as values for the x-axis.

The latter case is not often useful for standard dimensions (for example, SalesPerson). However, it is often used with the Time dimension.

After obtaining the set of points, **LinRegVariance** returns the statistical variance that describes the fit of the linear equation to the points.

Note Empty cells or cells containing text or logical values are ignored; however, cells with values of zero are included.

LookupCube

Returns the value of a Multidimensional Expressions (MDX) expression evaluated over another specified cube in the same database.

Syntax

Numeric

LookupCube(«Cube String», «Numeric Expression»)

The **LookupCube** function returns a numeric value, evaluating the numeric expression specified in «Numeric Expression» in another cube within the context of the cube specified in «Cube String».

String

LookupCube(«Cube String», «String Expression»)

The **LookupCube** function returns a string value, evaluating the string expression specified in «String Expression» in another cube within the context of the cube specified in «Cube String».

Remarks

The **LookupCube** function works only on other cubes within the same database. The function cannot be used to access cubes in a database other than the one established by the source cube context of the MDX expression. For example, if an MDX SELECT statement refers to the Sales cube in the **FoodMart 2000** database, the **LookupCube** function is limited to other cubes within the **FoodMart 2000** database.

Example

Numeric

The following example returns the value of the [Warehouse].[All Warehouses] member in the Warehouse cube:

```
LookupCube("Warehouse","[Warehouse].[All Warehouses]")
```

String

The following example returns the full name of the value of the [Warehouse].[Maddock Stored Foods] member in the Warehouse cube:

```
LookupCube("Warehouse", "MemberToStr([Warehouse].[Maddock Sto
```

M

Max

Returns the maximum value of a numeric expression evaluated over a set.

Syntax

Max(«Set»[, «Numeric Expression»])

Remarks

The **Max** function returns the maximum value of a numeric expression, specified in «Numeric Expression», evaluated from a set specified in «Set».

Example

The following example returns 3000 if respective Sales of the countries are 1000, 2000, and 3000:

Max({USA, CANADA, MEXICO}, Sales)

Median

Returns the median value of a numeric expression evaluated over a set.

Syntax

Median(«Set»[, «Numeric Expression»])

Remarks

The **Median** function returns the median value of a numeric expression, specified in «Numeric Expression», evaluated over a set specified in «Set».

Example

The following example returns 2000 if respective Sales of the countries are 1000, 2000, and 3000:

```
Median({USA, CANADA, MEXICO}, Sales)
```

Members

Returns the set of members in a dimension, level, or hierarchy. Alternatively, returns a member specified by a string expression.

Syntax

Dimension

«Dimension».**Members**

This syntax returns the set of all members in «Dimension».

Hierarchy

«Hierarchy».**Members**

This syntax returns the set of all members in «Hierarchy».

Level

«Level».**Members**

This syntax returns the set of all members at a specified level in a dimension.

String

Members(«String Expression»)

This syntax returns the member whose name is given by «String Expression» in Multidimensional Expressions (MDX) format. It is typically used with user-defined functions.

Examples

Dimension

This example

```
Geography.Members
```

returns the set of all members in the Geography dimension.

Hierarchy

This example

```
Time.Quarterly.Members
```

returns the set of all members in the Quarters hierarchy of the Time dimension.

Level

If the Year level contains [1994], [1995], and [1996], this example returns the set {[1994], [1995], [1996]}:

```
Year.Members
```

String

The following example returns a member, where UDF() is a user-defined function that returns a string in MDX format, such as "[Measures].[Unit Sales]":

```
Members(UDF())
```

MemberToStr

Returns a string in Multidimensional Expressions (MDX) format from a member.

Syntax

MemberToStr(«Member»)

Remarks

Returns a string, in MDX format, of the definition of a member. The **MemberToStr** function is typically employed for user-defined functions.

Example

The following example returns the string Time.[1998]:

```
MemberToStr(Time.[1998])
```

Min

Returns the minimum value of a numeric expression evaluated over a set.

Syntax

Min(«Set»[, «Numeric Expression»])

Remarks

The **Min** function returns the minimum value of a numeric expression, specified in «Numeric Expression», evaluated over a set specified in «Set».

Example

The following example returns 1000 if the respective Sales of the countries are 1000, 2000, and 3000:

```
Min({USA, CANADA, MEXICO}, Sales)
```

Mtd

Returns a set of members from the Month level in a Time dimension starting with the first period and ending with a specified member.

Syntax

Mtd([«Member»])

Remarks

The **Mtd** function is a shortcut function to the **PeriodsToDate** function that defines that function's «Level» argument to be Month. If no member is specified, the default is **Time.CurrentMember**.

Mtd(«Member») is equivalent to **PeriodsToDate**(Month, «Member»).

Example

The following example returns the first five days of September, 1997:

```
MTD([05-Sep-1997])
```

N

Name

Returns the name of a level, dimension, member, or hierarchy.

Syntax

Dimension

«Dimension».**Name**

Level

«Level».**Name**

Member

«Member».**Name**

Hierarchy

«Hierarchy».**Name**

Remarks

The **Name** function returns the name of the object, not the unique name.

Examples

Dimension

```
Products.Name
```

Level

```
Products.[Product Description].Name
```

Member

Products.[Product Description].Widgets.Name

Hierarchy

Time.[Fiscal Year].Name

NameToSet

Returns a set containing a single member based on a string expression containing a member name.

Syntax

NameToSet(«Member Name»)

Remarks

If the member name specified in «Member Name» exists, a set containing that member is returned. Otherwise, the **NameToSet** function returns an empty set.

Example

The following function returns a set containing the [Quarter 1] member of the Time dimension:

```
NameToSet("[Time].[Quarter 1]")
```

NextMember

Returns the next member in the level that contains a specified member.

Syntax

«Member».**NextMember**

Remarks

The **NextMember** function returns the next member in the same level that contains the member specified in «Member».

Example

If the Year level consists of members named [1994], [1995], and [1996], the following example returns [1995]:

[1994].NextMember

NonEmptyCrossjoin

Returns the cross product of two or more sets as a set, excluding empty tuples and tuples without associated fact table data.

Syntax

NonEmptyCrossjoin(«Set1», «Set2»[, «Set3»...][, «Crossjoin Set Count»])

Remarks

The **NonEmptyCrossjoin** function returns the cross product of two or more sets as a set, excluding empty tuples or tuples without data supplied by underlying fact tables; because of this, all calculated members are automatically excluded. If «Crossjoin Set Count» is not specified, all specified sets are crossjoined and empty members are excluded from the resulting set. If «Crossjoin Set Count» is specified, the number of sets specified in «Crossjoin Set Count», starting with «Set1», are crossjoined. The remaining sets are used to determine, in the resulting crossjoined set, which members are considered nonempty.

For example, you want to view the total unit sales for each store in Beverly Hills involved with the Big Time Savings promotion, but only for those customers based in California. However, the following Multidimensional Expressions (MDX) statement returns a set containing the unit sales for all the cities in California with customers, grouped by the stores in Beverly Hills, involved with the Big Time Savings promotion; this set is a cross product of the three sets. The unit sales returned are only for the cross product of the three sets; the gross sales of the stores in Beverly Hills involved with the Big Time Savings promotion are not returned, but the individual unit sales involved with just the promotion itself, for each store and customer city, are.

```
NonEmptyCrossJoin([Store].[Beverly Hills].Children, [Customers].[C
```

The previous example was too narrow in scope to accomplish the task. By contrast, the following MDX statement simply crossjoins the first two sets and removes nonempty members from the returned set. Because the {[Promotions].

[Big Time Savings]} set is not used, the preceding MDX statement is too wide in scope; the MDX statement includes too many tuples to accomplish the goal.

```
NonEmptyCrossJoin([Store].[Beverly Hills].Children, [Customers].[C
```

The following MDX statement, by using the «Crossjoin Set Count» parameter, returns a set containing the unit sales for all of the cities in California with customers grouped by the stores in Beverly Hills; this set is a cross product of the first two sets. Only those members in the crossjoined set that participated in the Big Time Savings promotion are returned, however, accomplishing the task. The first two sets, specified in the «Crossjoin Set Count» parameter, were crossjoined, while the third set was used to determine which members of the crossjoined set were to be considered when determining if a crossjoin set member contained data.

```
NonEmptyCrossJoin([Store].[Beverly Hills].Children, [Customers].[C
```

The benefits of the **NonEmptyCrossjoin** function include faster, more efficient processing of crossjoins involving more than two sets, as well as the simpler syntax provided by the function. The same results can be obtained less effectively, using the **Filter**, **Crossjoin**, and **IsEmpty** functions as shown in the following MDX statement:

```
Filter(Crossjoin([Store].[Beverly Hills].Children, [Customers].[CA].Cl
```

As additional sets are added, the use of **Filter**, **Crossjoin**, and **IsEmpty** becomes increasingly impractical, because each **Crossjoin** statement is nested inside another **Crossjoin** statement to return the same results. For example, adding the [Product].Children set to the returned set using the **NonEmptyCrossjoin** function resembles:

```
NonEmptyCrossJoin([Store].[Beverly Hills].Children, [Customers].[C
```

Performing the same functionality with the **Filter**, **Crossjoin**, and **IsEmpty** functions, on the other hand, resembles the following:

```
Filter(Crossjoin(Crossjoin([Store].[Beverly Hills].Children, [Customer
```

The preceding MDX statement is slower in execution and less readable than its

NonEmptyCrossjoin-based counterpart.

Example

The following statement returns the set containing all the unit sales for all of the cities in California with customers grouped by the stores in Beverly Hills that participated in the Big Time Savings promotion:

```
NonEmptyCrossJoin([Store].[Beverly Hills].Children, [Customers].[C
```

See Also

Crossjoin

Filter

IsEmpty

O

OpeningPeriod

Returns the first sibling among the descendants of a specified level, optionally at a specified member.

Syntax

OpeningPeriod([«Level»[, «Member»]])

Remarks

The dimension that contains «Level» is used if «Level» is specified; otherwise, the Time dimension is used. If no «Level» is specified, the level below that of «Member» is used. If no «Level» or «Member» is specified, the default is Time.**CurrentMember**.

This function is equivalent to **TopCount(Descendants(**«Member»**,** «Level»**)**, 1**)**.

The **ClosingPeriod** function is similar, except that it returns the last sibling instead of the first sibling.

Example

The following example returns [1991].January:

OpeningPeriod(Month, [1991])

Order

Arranges members of a specified set, optionally preserving or breaking the hierarchy.

Syntax

Order(«Set», {«String Expression» | «Numeric Expression»}
[, ASC | DESC | BASC | BDESC])

Remarks

There are two varieties of **Order**: hierarchized (ASC or DESC) and nonhierarchized (BASC or BDESC, where B stands for Break hierarchy). The hierarchized ordering first arranges members according to their position in the hierarchy. Then it orders each level. The nonhierarchized ordering arranges members in the set without regard to the hierarchy. In the absence of an explicit specification, ASC is the default.

Example

This example

```
Order(SampleSet, ([1995], Sales), DESC)
```

hierarchizes all members and sorts each level according to Sales. Sales are compared at the highest level when the sorted list is constructed. Therefore, if the sum of Sales in all California cities is less than the sum of Sales in all New York cities, California and California.LA will appear below NYC in the sorted, descending list.

The result of

```
Order(SampleSet, ([1995], Sales), DESC)
```

is listed in the following table.

Location			1995 sales
USA			5000
	California		2000
		LA	500
		Buffalo	300
		NYC	900
France			2500
		Paris	365
		Nice	27
UK			1900
		London	250

The following expression sorts the members according to their values without regard for their relative positions in the member hierarchy. In this example, numeric values are sorted by 1995 sales per city, including aggregate sales values by state and country:

```
Order(SampleSet, ([1995], Sales), BDESC)
```

The following table shows the result of the previous expression.

Location	1995 sales
USA	5000
France	2500
California	2000
UK	1900
NYC	900
LA	500
Paris	365
Buffalo	300
London	250
Nice	27

Note When the input set has two elements for which the «String Expression» or «Numeric Expression» has the same value, the input order is preserved.

For example, if the sales for USA and Europe is 300 each, and the sales for Asia is 100, the following expression returns the set {Asia, USA, Europe}, not the set {Asia, Europe, USA}:

```
Order({USA, Europe, Asia}, Sales, BASC)
```

Ordinal

Returns the zero-based ordinal value associated with a level.

Syntax

«Level».**Ordinal**

Example

For a dimension named Products with three levels, named [All Products], [Product Category], and [Product Name], the following example returns 2:

```
Products.[Product Name].Ordinal
```

P

ParallelPeriod

Returns a member from a prior period in the same relative position as a specified member.

Syntax

ParallelPeriod([«Level»[, «Numeric Expression»[, «Member»]]])

Remarks

This function is similar to the **Cousin** function, but is more closely related to time series. It takes the ancestor of «Member» at «Level» (call it *ancestor*); then it takes the sibling of *ancestor* that lags by «Numeric Expression», and returns the parallel period of «Member» among the descendants of that sibling.

This function has the following defaults:

- Default «Member» value is **Time.CurrentMember** if «Level» is not specified. Otherwise it is *dimension***.CurrentMember**, where *dimension* is the dimension to which *level* belongs.

- Default «Numeric Expression» is 1.

- Default «Level» is the level of the parent of «Member».

This function is equivalent to **Cousin(**Member**,Lag(Ancestor(**Member,Level**),**Numeric Expression**)**.

Example

The following table lists different examples of how you can use this function.

Expression	Returns
ParallelPeriod(Year,2,	[94 Qtr 3]

[96 Qtr 3])	
ParallelPeriod(Year,2)	The parallel period of Time.**CurrentMember**, two years ago. That is, if Time.**CurrentMember** is [1993June], the returned member is [1991June].
ParallelPeriod(Year)	The parallel period of Time.**CurrentMember** from last year. That is, if Time.**CurrentMember** is [1993June], the returned member is [1992June].
ParallelPeriod()	The parallel period in the immediately prior sibling to the parent of **Time.CurrentMember**. For example, if **Time.CurrentMember** is [1993June], the returned member is [1993March]. The parent of ([1993June] is Quarter2, whose immediately prior sibling is Quarter1, in which the parallel period is [1993March].

Parent

Returns the parent of a member.

Syntax

«Member».**Parent**

Remarks

This function returns the parent member of the member specified in «Member».

Example

If the Geography dimension includes State and Country, the following example returns `USA`:

```
California.Parent
```

PeriodsToDate

Returns a set of periods (members) from a specified level starting with the first period and ending with a specified member.

Syntax

PeriodsToDate([«Level»[, «Member»]])

Remarks

Within the scope of «Level», this function returns the set of periods on the level of «Member», starting with the first period and ending with «Member». If no level or member is specified, the «Member» value is Time.**CurrentMember** and «Level» is the parent level of Time.**CurrentMember**. If a level is specified, «Member» is *dimension***.CurrentMember**, where *dimension* is the dimension of «Level».

Example

The following table lists the different ways you can use **PeriodsToDate**.

Expression	Returns
PeriodsToDate(Quarter, [05-Sep-1997])	The set of days from the beginning of Quarter3.
PeriodsToDate(Year, March)	The set {January, February, March}.
PeriodsToDate(Year)	The set of members from the beginning of the year that is the ancestor of Time.**CurrentMember**, through Time.**CurrentMember**.
PeriodsToDate()	The set of members from the beginning of the level containing the period of Time.**CurrentMember** to

	Time.**CurrentMember**. All the returned members are at the same level as Time.**CurrentMember**.

PeriodsToDate(*level*, *member*) is the same as **TopCount**(**Descendants**(**Ancestor**(*member*, *level*), *member***.Level**), 1):*member*

Predict

Returns a value of a numeric expression evaluated over a data mining model.

Syntax

Predict(«Mining Model Name», «Numeric Expression»)

Remarks

The **Predict** function evaluates the numeric expression specified in «Numeric Expression» in another data mining model, within the context of the mining model specified in «Mining Model Name».

Data mining syntax and functions are documented in the OLE DB for Data Mining specification. For more information about the OLE DB for Data Mining specification, see the Microsoft OLE DB Web page at the Microsoft Web site.

Example

The following code returns the variance value for the histogram entry in the [Gender] data mining column that has the highest probability in the Customer Pattern Discovery data mining model:

```
Predict("Customer Pattern Discovery","PredictVariance([Customer Pat
```

PrevMember

Returns the previous member in the level that contains a specified member.

Syntax

«Member».**PrevMember**

Remarks

This function returns the previous member in the same level of the member specified in «Member».

Example

If the Year level consists of [1994], [1995], and [1996], the following example returns the member [1995]:

```
[1996].PrevMember
```

Properties

Returns a string containing a member property value.

Syntax

«Member».**Properties**(«String Expression»)

Remarks

The **Properties** function returns the value of the member property specified in «String Expression». The member property can be any of the standard member properties, such as NAME, ID, KEY, or CAPTION, or it can be a user-defined member property.

Example

In the Store dimension, if the Store Name level has an associated member property, Store Manager, the following example returns Smith:

```
[Store].[All Stores].[USA].[WA].[Bellingham].[Store 2].Properties("St
```

Q

Qtd

Returns a set of members from the Quarter level in a Time dimension starting with the first period and ending with a specified member.

Syntax

Qtd([«Member»])

Remarks

This shortcut function to the **PeriodsToDate** function predefines that function's «Level» argument to be Quarter. If no member is specified, the default is **Time.CurrentMember**.

Qtd(«Member») is equivalent to **PeriodsToDate**(Quarter, «Member»).

Example

The following example returns the set of days from the beginning of the third quarter of 1997:

```
Qtd([05-Sep-1997]))
```

See Also

PeriodsToDate

R

Rank

Returns the one-based rank of a specified tuple in a specified set.

Syntax

Rank(«Tuple», «Set»[, «Calc Expression»])

Remarks

If «Calc Expression» is not specified, the **Rank** function returns the one-based ordinal position of a tuple, specified in «Tuple», within a set specified in «Set».

If «Calc Expression» is specified, the **Rank** function evaluates the numeric expression specified in «Calc Expression» against the tuple to determine its one-based rank. When «Calc Expression» is specified, the **Rank** function assigns the same rank to tuples in a set with duplicate values. However, the presence of duplicate values affects the ranks of subsequent tuples in the set. For example, if the tuple (a,b) had the same value as the tuple (c,d) in the set {(a,b), (e,f), (c,d)}, and the tuple (a,b) has a rank of 1, then (a,b) and (c,d) would both have a rank of 1, but (e,f) would have a rank of 3. No tuple would have a rank of 2 in the set.

The **Rank** function does not order the set.

Example

The following example returns 3:

```
Rank((c,d), {(a,b), (e,f), (c,d)})
```

However, if the tuples in the set { (a,b), (e,f), (c,d) } have values of 1, 8, and 3, respectively, in the [Test] measure, the following example returns 2:

```
Rank ((c,d), {(a,b), (e,f), (c,d)}, Measures.Test)
```

RollupChildren

Returns a value generated by rolling up the values of the children of a specified member using the specified unary operator.

Syntax

RollupChildren(«Member», «String Expression»)

Remarks

The **RollupChildren** function rolls up the values of the children of the member specified in «Member», using the unary operator specified in «String Expression».

The following table describes the valid unary operators for this function.

Operator	Result
+	total = total + current child
-	total = total - current child
*	total = total * current child
/	total = total / current child
%	total = (total / current child) * 100
~	Child is not used in the rollup. Its value is ignored.

If the operator in the member property does not appear in the list, an error occurs. The order of evaluation is determined by the order of the siblings (not by the precedence of the operators).

Example

The following example uses the default unary operator for each child member to create the rollup for the parent member:

```
RollupChildren(Location.CurrentMember, Location.CurrentMember.P
```

S

SetToArray

Converts one or more sets to an array for use in a user-defined function.

Syntax

SetToArray(«Set»[, «Set»...][, «Numeric Expression»])

Remarks

This function converts one or more sets to an array for use in a user-defined function. The number of dimensions in the resulting array is the same as the number of sets specified.

The optional numeric expression can be used to provide the values in the array cells. If omitted, the default value of the set member is used for the array cell value.

The cell coordinates in the resulting array correspond to the position of the sets in the list. For example, **SetToArray**(SA, SB, SC), where each set has two elements, produces the three-dimensional array:

```
(SA1, SB1, SC1) (SA2, SB1, SC1) (SA1, SB2, SC1) (SA2, SB2, SC1)
(SA1, SB1, SC2) (SA2, SB1, SC2) (SA1, SB2, SC2) (SA2, SB2, SC2)
```

Note The return type of **SetToArray** is the VARIANT type VT_ARRAY. For that reason, the output of **SetToArray** should be used only as input to a user-defined function.

Example

```
SetToArray(Geography.Members, Measures.Sale)
```

SetToStr

Constructs a string in Multidimensional Expressions (MDX) format from a set.

Syntax

SetToStr(«Set»)

Example

The following example returns "{[Time].[1995], [Time].[1996]}":

```
SetToStr({1995, 1996})
```

Siblings

Returns the siblings of a specified member, including the member itself.

Syntax

«Member».**Siblings**

Examples

The following example returns the set { January, February, March }:

[Time].[All Time].[1998].[Quarter 1].[January].Siblings

Stddev

Alias for **Stdev**.

See Also

Stdev

StddevP

Alias for **StdevP**.

See Also

StdevP

Stdev

Returns the sample standard deviation of a numeric expression evaluated over a set, using the unbiased population formula.

Syntax

Stdev(«Set»[, «Numeric Expression»])

Remarks

The **Stdev** function uses the unbiased population formula, while the **StdevP** function uses the biased population formula.

Example

```
Stdev({USA, CANADA, MEXICO}, Sales)
```

See Also

StdevP

StdevP

Returns the population standard deviation of a numeric expression evaluated over a set, using the biased population formula.

Syntax

StdevP(«Set»[, «Numeric Expression»])

Remarks

The **StdevP** function uses the biased population formula, while the **Stdev** function uses the unbiased population formula.

Example

Stdev({USA, CANADA, MEXICO}, Sales)

See Also

Stdev

StripCalculatedMembers

Returns a set generated by removing calculated members from a specified set.

Syntax

StripCalculatedMembers(«Set»)

Remarks

This function removes calculated members from a set that includes calculated members added using **AddCalculatedMembers**.

Example

```
StripCalculatedMembers([Store Size in SQFT].[All Store Size in SQF
```

StrToMember

Returns a member from a string expression in Multidimensional Expressions (MDX) format.

Syntax

StrToMember(«String Expression»)

Returns a member from a string in MDX format containing a member, specified in «String Expression».

Remarks

The **StrToMember** function is typically employed with user-defined functions.

Example

The following example returns the member Time.[1996]:

```
StrToMember("Time.[1996]")
```

StrToSet

Constructs a set from a specified string expression in Multidimensional Expressions (MDX) format.

Syntax

StrToSet(«String Expression»)

Example

The following example returns a set containing all the members of the Time dimension:

```
StrToSet("Time.Members")
```

StrToTuple

Constructs a tuple from a specified string expression in Multidimensional Expressions (MDX) format.

Syntax

StrToTuple(«String Expression»)

Example

The following example returns (Sales, [1994]):

```
StrToTuple("(Sales, [1994])")
```

StrToValue

Returns a value from a string expression.

Syntax

StrToValue(«String Expression»)

Remarks

The **StrToValue** function is typically used for user-defined functions.

Example

The following example returns 45:

```
StrToValue("45")
```

Subset

Returns a subset of tuples from a specified set.

Syntax

Subset(«Set», «Start»[, «Count»])

Remarks

This function returns «Count» tuples from «Set» as a set, starting at position «Start». «Start» is zero-based: 0 corresponds to the first tuple in the set, 1 corresponds to the second, and so on. If «Count» is not specified, all tuples from «Start» to the end of the set are returned.

Example

The following example returns the set {USA, Canada}:

```
Subset({USA, Canada, France, Germany, Japan, England, Peru}, 0, 2)
```

Sum

Returns the sum of a numeric expression evaluated over a set.

Syntax

Sum(«Set»[, «Numeric Expression»])

Example

If the respective values for the USA, CANADA and MEXICO members are 1000, 2000, and 3000, the following example returns 6000:

```
Sum({USA, CANADA, MEXICO}, Sales.VALUE)
```

The following example, which is more intuitive, is also valid:

```
Sum({USA, CANADA, MEXICO}, Sales)
```

T

Tail

Returns a subset from the end of a set.

Syntax

Tail(«Set»[, «Count»])

Remarks

This function returns the last «Count» elements in a set. The order of elements is preserved. The default value of «Count» is 1. If «Count» is less than 1, the empty set is returned. If «Count» exceeds the number of tuples in the set, the original set is returned.

Example

The following code returns the set {France, Germany, Japan}:

```
Tail({USA, Canada, France, Germany, Japan}, 3)
```

ToggleDrillState

Toggles the drill state of members.

Syntax

ToggleDrillState(«Set1», «Set2»[, RECURSIVE])

Remarks

This function is a combination of **DrillupMember** and **DrilldownMember**. It toggles the drill state of each member of «Set2» that is present in «Set1». If a member *m* of «Set2» that is present in «Set1» is drilled down (that is, has a descendant), **DrillupMember(**«Set1»**,** {*m*}**)** is applied. If it is drilled up (that is, there is no descendant of *m* that immediately follows *m*), **DrilldownMember(**«Set1»**,** {*m*}[, RECURSIVE]**)** is applied to *Set1*. The optional RECURSIVE flag is used if **ToggleDrillState** was called with RECURSIVE.

Example

```
ToggleDrillState({Product.Bread.Members},{Product.Bagels, Product.
```

See Also

DrillupMember

DrilldownMember

TopCount

Returns a specified number of items from the topmost members of a specified set, optionally ordering the set first.

Syntax

TopCount(«Set», «Count»[, «Numeric Expression»])

Remarks

This function sorts a set according to the value of «Numeric Expression» and returns the top «Count» members, where «Count» is a numeric expression.

IMPORTANT Like the **BottomCount** function, this function always breaks the hierarchy.

Example

```
Topcount(Geography.Cities.Members, 5, Sales)
```

See Also

BottomCount

TopPercent

Sorts a set and returns the topmost elements whose cumulative total is at least a specified percentage.

Syntax

TopPercent(«Set», «Percentage», «Numeric Expression»)

Remarks

This function sorts a set using «Numeric Expression» and returns the top *n* elements whose cumulative total of «Numeric Expression» is at least «Percentage». «Percentage» is a numeric expression.

Important Like the **BottomPercent** function, this function always breaks the hierarchy.

Example

```
TopPercent({London, Paris, Rome, New York, Seattle, Tokyo}, 15, Sa
```

See Also

BottomPercent

TopSum

Sorts a set and returns the topmost elements whose cumulative total is at least a specified value.

Syntax

TopSum(«Set», «Value», «Numeric Expression»)

Remarks

This function sorts on «Numeric Expression» and picks up the top *n* (the smallest number possible) elements such that their sum is at least «Value».

IMPORTANT Like the **BottomSum** function, this function always breaks the hierarchy.

Example

```
Topsum(Products.[Product Description].Members, 100000, Quantity)
```

See Also

BottomSum

TupleToStr

Returns a string in Multidimensional Expressions (MDX) format from a specified tuple.

Syntax

TupleToStr(«Tuple»)

Example

The following example returns "([Time].[1995], [Regions].[All Regions].[Europe].[France])":

```
TupleToStr(France, 1995)
```

U

Union

Returns a set generated by the union of two sets, optionally retaining duplicate members.

Syntax

Union(«Set1», «Set2»[, ALL])

Alternate Syntax 1

{«Set1», «Set2»}

Alternate Syntax 2

«Set1» + «Set 2»

Remarks

This function returns the union of «Set1» and «Set2», eliminating duplicates by default. The ALL flag keeps duplicates in the joined set. Duplicates are deleted from the tail.

You can also join by union using either a comma-separated list of sets within braces or the + operator. For example:

```
{USA.Children, CANADA.Children}
```

and

```
{USA.Children} + {CANADA.Children}
```

are equivalent to

```
Union(USA.Children, CANADA.Children, ALL)
```

Duplicated members are always retained when an alternate syntax is used.

Note This alternate syntax is a Microsoft® SQL Server™ 2000 Analysis Services extension to Multidimensional Expressions (MDX) in OLE DB.

UniqueName

Returns the unique name of a specified level, dimension, member, or hierarchy.

Syntax

Dimension

«Dimension».**UniqueName**

Level

«Level».**UniqueName**

Member

«Member».**UniqueName**

Hierarchy

«Hierarchy».**UniqueName**

Remarks

The **UniqueName** function returns the unique name of the object, not the name.

Examples

Dimension

```
Products.UniqueName
```

Level

```
Products.[Product Description].UniqueName
```

Member

Products.[Product Description].Widgets.UniqueName

Hierarchy

Time.[Fiscal Year].UniqueName

UserName

Returns the domain name and user name of the current connection.

Syntax

UserName

Remarks

The returned value has the following format:

domain-name\user-name

Example

If a user is connected to the NYC domain, and the name of the user is alanc, the following example returns NYC\alanc:

```
UserName
```

V

ValidMeasure

Returns a valid measure in a virtual cube by forcing inapplicable dimensions to their top level.

Syntax

ValidMeasure(«Tuple»)

Returns a valid measure from a tuple in a virtual cube specified in «Tuple».

Remarks

When computing values in a virtual cube, measure cells only contain values at the (All) level for dimensions that are not common between the underlying cubes. The **ValidMeasure** function returns the measure value from the cell at the (All) level coordinates for the dimensions that are not common.

Example

Consider a virtual cube named ExchSales that is derived from cubes Sales and ExchRate. The Sales cube contains dimensions Time, Products, and Customers, and the measure Sale in dollars. The ExchRate cube contains the dimensions Time and Currency and the measure Rate.

To create a calculated member named YenSale that returns the value of each sale in Yen, the (All) level must be specified for each dimension not present in the ExchRate cube for there to be a value present in the virtual cube cell. This can be cumbersome for virtual cubes where a number of dimensions are not common between the underlying cubes.

The **ValidMeasure** function ensures the measure's value is taken from the appropriate cell at the (All) level for each dimension not in common. The following two Multidimensional Expressions (MDX) statements are equivalent:

```
YenSale = Sum({Descendants(Time.CurrentMember, Day}},
    Sale * (Currency.Yen, AllProducts, AllCustomers))
```

```
YenSale = Sum({Descendants(Time.CurrentMember, Day}},
    Sale * ValidMeasure(Currency.Yen))
```

Value

Returns the value of a measure.

Syntax

«Member».**Value**

Remarks

The **Value** function returns the value of a measure specified in «Member». This is the default property of a measure.

Example

```
Sales.Measures.CurrentMember.Value
```

Var

Returns the sample variance of a numeric expression evaluated over a set, using the unbiased population formula.

Syntax

Var(«Set»[, «Numeric Expression»])

Remarks

The **Var** function returns the unbiased variance of a numeric expression, specified in «Numeric Expression», evaluated over a set specified in «Set».

The **Var** function uses the unbiased population formula, while the **VarP** function uses the biased population formula.

Example

```
Var ({USA, CANADA, MEXICO}, Sales)
```

See Also

VarP

Variance

Alias for **Var**.

See Also

Var

VarianceP

Alias for **VarP**.

See Also

VarP

VarP

Returns the population variance of a numeric expression evaluated over a set, using the biased population formula.

Syntax

VarP(«Set»[, «Numeric Expression»])

Remarks

The **VarP** function returns the biased variance of a numeric expression, specified in «Numeric Expression», evaluated over a set specified in «Set».

The **VarP** function uses the biased population formula, while the **Var** function uses the unbiased population formula.

Example

```
VarP({USA, CANADA, MEXICO}, Sales)
```

See Also

Var

VisualTotals

Returns a set generated by dynamically totaling child members in a specified set, optionally using a pattern for the name of the parent member in the result set.

Important The **VisualTotals** function cannot be used in a query to a cube that contains a distinct count measure; such a query will return an error for all measure values. For more information, see Using Aggregate Functions.

Syntax

VisualTotals(«Set», «Pattern»)

Remarks

This function totals the values of the child members specified in «Set» only. Child members not specified in «Set» will not be included in the result. «Pattern» specifies the format for the totals label. Text for the pattern is taken literally and the asterisk (*) is the substitution character for the parent member. To display a literal asterisk, use two asterisks (**).

Note The **VisualTotals** function replaces the parent member of the resulting cellset. Multiple hierarchies of Parent and Child members may be specified in «Set».

Example

Assume that the Product dimension has the member Baked Goods with a child of Bread. Bread has the child members Bagels, Muffins, and Sliced Bread.

In the first case, a select statement is done using

```
[Product].[All Products].[Food].[Baked Goods].[Bread]
```

as the parent member and

```
[Product].[All Products].[Food].[Baked Goods].[Bread].[Bagels]
```

```
[Product].[All Products].[Food].[Baked Goods].[Bread].[Muffins]
```

for the child members. The results for the parent member reflect the precalculated values of all of its children and do not take into account that other child members have not been included in the resulting set.

```
select
  {[Measures].[Unit Sales]} on columns,
  {[Product].[All Products].[Food].[Baked Goods].[Bread],
   [Product].[All Products].[Food].[Baked Goods].[Bread].[Bagels],
   [Product].[All Products].[Food].[Baked Goods].[Bread].[Muffins]
  } on rows
from Sales
```

	Unit Sales
Bread	7,870.00
Bagels	815.00
Muffins	3,497.00

An alternate solution is to use the **VisualTotals** function to dynamically total the child members in the set and display an accurate value for Bread.

```
select
{[Measures].[Unit Sales]} on columns,
{VisualTotals({[Product].[All Products].[Food].[Baked Goods].[Bread
  [Product].[All Products].[Food].[Baked Goods].[Bread].[Bagels],
  [Product].[All Products].[Food].[Baked Goods].[Bread].[Muffins]},
   "**Subtotal - *")
} on rows
from Sales
```


	Unit Sales
*Subtotal - Bread	4,312.00
Bagels	815.00
Muffins	3,497.00

The string "*Subtotal - Bread" is constructed by substituting the single asterisk substitution character with "Bread" to produce a meaningful name for the dynamically calculated total. The double asterisks in the substitution string specify the output asterisk in the string "*Subtotal - Bread".

W

Wtd

Returns a set of members from the Week level in a Time dimension starting with the first period and ending with a specified member.

Syntax

Wtd([«Member»])

Remarks

The **Wtd** function is a shortcut function to the **PeriodsToDate** function that defines that function's «Level» argument to be Week. If no member is specified, then the default is Time.**CurrentMember**.

Wtd(«Member») is equivalent to **PeriodsToDate**(Week, «Member»).

Example

The following example returns the days from the beginning of the week to the current day:

```
Wtd(Day)
```

See Also

PeriodsToDate

Y

Ytd

Returns a set of members from the Year level in a Time dimension starting with the first period and ending with a specified member.

Syntax

Ytd([«Member»])

Remarks

The **Ytd** function is a shortcut function to the **PeriodsToDate** function that defines that function's «Level» argument to be Year. If no member is specified, the default is **Time.CurrentMember**.

Ytd(«Member») is equivalent to **PeriodsToDate**(Year, «Member»).

Example

The following example returns the set of members from the beginning of the year from the ancestor of **Time.CurrentMember** through **Time.CurrentMember**:

```
Ytd()
```

See Also

PeriodsToDate

Analysis Services

Registered Function Libraries

Microsoft® SQL Server™ 2000 Analysis Services includes and automatically registers the Microsoft Visual Basic® for Applications Expression Services library of functions, and automatically registers the Microsoft Excel worksheet library if it is installed on the computer with Analysis Services.

Analysis Services supports many but not all functions in these libraries. For information about supported functions, see Visual Basic for Applications Functions and Excel Functions.

Visual Basic for Applications Functions

Microsoft® SQL Server™ 2000 Analysis Services supports many functions in the Microsoft Visual Basic® for Applications Expression Services library. This library is included with Analysis Services and automatically registered. Functions not supported in this release are marked by an asterisk in this table.

For more information about syntax and examples of these functions, search on the function name in the MSDN® Library at the Microsoft Web site.

Abs	*Add	*AppActivate	Array
Asc	AscB	AscW	Atn
*Beep	*Calendar	*CallByName	CBool
CByte	Ccur	CDate	CDbl
*CDec	*ChDir	*ChDrive	Choose
Chr	*ChrB	ChrW	CInt
*Clear	CLng	*Command	Cos
*Count	*CreateObject	CSng	CStr
*CurDir	Cvar	CVDate	*CVErr
Date	DateAdd	DateDiff	DatePart
DateSerial	DateValue	Day	DDB
*DeleteSetting	*Description	*Dir	*DoEvents
*Environ	*EOF	*Err	*Error
Exp	*FileAttr	*FileCopy	*FileDateTime
FileLen	*Filter	Fix	Format
*FormatCurrency	*FormatDateTime	*FormatNumber	*FormatPercent
*FreeFile	FV	*GetAllSettings	*GetAttr
*GetObject	*GetSetting	*HelpContext	*HelpFile
Hex	Hour	IIf	*IMEStatus
*Input	*InputB	*InputBox	InStr
InStrB	*InStrRev	Int	IPmt
*IRR	*IsArray	IsDate	IsEmpty

IsError	*IsMissing	IsNull	IsNumeric
IsObject	*Item	*Join	*Kill
*LastDllError	LCase	Left	LeftB
Len	LenB	*Loc	*LOF
Log	LTrim	Mid	MidB
Minute	*MIRR	*MkDir	Month
*MonthName	*MsgBox	Now	NPer
*NPV	*Number	Oct	Partition
Pmt	PPmt	PV	QBColor
*Raise	*Randomize	Rate	*Remove
*Replace	*Reset	RGB	Right
RightB	*RmDir	Rnd	Round
RTrim	*SaveSetting	Second	*Seek
*SendKeys	*SetAttr	Sgn	*Shell
Sin	SLN	*Source	Space
*Split	Sqr	Str	StrComp
String	*StrReverse	Switch	SYD
Tan	Time	Timer	TimeSerial
TimeValue	Trim	TypeName	UCase
Val	*VarType	Weekday	*WeekdayName
*Width	Year		

Excel Functions

Microsoft® SQL Server™ 2000 Analysis Services supports many functions in the Microsoft Excel worksheet library, which is automatically registered if installed on the computer with Analysis Services. Functions not supported in this release are marked by an asterisk in this table.

Acos	Acosh	And	*Application
Asc	Asin	Asinh	Atan2
Atanh	AveDev	Average	BetaDist
BetaInv	BinomDist	Ceiling	ChiDist
ChiInv	ChiTest	Choose	Clean
Combin	Confidence	Correl	Cosh
Count	CountA	*CountBlank	*CountIf
Covar	*Creator	CritBinom	*DAverage
Days360	Db	Dbcs	*DCount
*DCountA	Ddb	Degrees	DevSq
*DGet	*DMax	*DMin	Dollar
*DProduct	*DStDev	*DStDevP	*DSum
*DVar	*DVarP	Even	ExponDist
Fact	FDist	Find	FindB
FInv	Fisher	FisherInv	Fixed
Floor	Forecast	*Frequency	FTest
Fv	GammaDist	GammaInv	GammaLn
GeoMean	*Growth	HarMean	*HLookup
HypGeomDist	*Index	Intercept	Ipmt
Irr	IsErr	IsError	IsLogical
IsNA	IsNonText	IsNumber	Ispmt
IsText	Kurt	Large	*LinEst
Ln	Log	Log10	*LogEst
LogInv	LogNormDist	*Lookup	Match
Max	*MDeterm	Median	Min

*MInverse	MIrr	*MMult	Mode
NegBinomDist	NormDist	NormInv	NormSDist
NormSInv	NPer	Npv	Odd
Or	*Parent	Pearson	Percentile
PercentRank	Permut	Pi	Pmt
Poisson	Power	Ppmt	Prob
Product	Proper	Pv	Quartile
Radians	*Rank	Rate	Replace
ReplaceB	Rept	Roman	Round
RoundDown	RoundUp	RSq	Search
SearchB	Sinh	Skew	Sln
Slope	Small	Standardize	StDev
StDevP	StEyx	Substitute	*Subtotal
Sum	*SumIf	SumProduct	SumSq
SumX2MY2	SumX2PY2	SumXMY2	Syd
Tanh	TDist	Text	TInv
Transpose	*Trend	Trim	TrimMean
TTest	USDollar	Var	VarP
Vdb	*VLookup	Weekday	Weibull
ZTest			

Analysis Services

User-Defined Functions with MDX Syntax

You can create and register your own functions that operate on multidimensional data. These functions, called user-defined functions, can accept arguments and return values in the Multidimensional Expressions (MDX) syntax. You can create user-defined functions using Component Object Model (COM) automation languages such as Microsoft® Visual Basic® or Microsoft Visual C++®. A user-defined function can be developed using any tool capable of generating Microsoft ActiveX® libraries.

Before you use a user-defined function, you must register the library (that is, file) in which it is compiled. You can register user-defined function libraries of the following types:

- Type libraries (*.olb, *.tlb, *.dll)
- Executable files (*.exe, *.dll)
- ActiveX controls (*.ocx)

To register a user-defined function library, issue a USE LIBRARY statement. Its syntax is:

```
USE LIBRARY "<library_path_and_file_name>" | <program_ID>
[,"<library_path_and_file_name>" | <program_ID>...]
```

Example:

```
USE LIBRARY "c:\functions\mylib.dll"
```

To register multiple libraries, issue a USE LIBRARY statement with multiple parameters in a comma-separated list. Example:

```
USE LIBRARY "c:\functions\mylib.dll","c:\functions\johnslib.dll"
```

A USE LIBRARY statement with no parameters unregisters all function libraries except the Microsoft SQL Server™ 2000 Analysis Services function library.

Hidden and restricted user-defined functions are not supported.

Note User-defined functions are supported only if they accept as arguments only string or numeric data types, or array or variant data types containing string or numeric values. In addition, user-defined functions are supported only if they return only string or numeric data types, or variant data types containing numeric values.

Multiple user-defined functions can reside in the same ActiveX library.

Calling a User-Defined Function within MDX

After a user-defined function is registered, it can be used anywhere in the MDX syntax that allows expressions. For example:

```
With Member Measures.[Forecasted Sales] As
    'Sales * ForecastedGrowthRate(SaleReps.CurrentMember.Name)'
Select TopCount(SalesReps, HowManyReps(), Sales) on Rows,
    {Sales, [Forecasted Sales] } on Columns
From Sales
```

The HowManyReps and ForecastedGrowthRate user-defined functions are defined as:

```
Public Function HowManyReps() as Integer
Public Function ForecastedGrowthRate(RepName as String) as Doubl
```

User-defined functions can also be used in Calculated Member Builder.

Note When you call a user-defined function, you can omit an optional argument only if you also omit all arguments that follow it.

Function Precedence and Qualification

If multiple function libraries contain a function with the same name, the Analysis Services function library takes precedence. Excluding the Analysis Services function library, precedence is resolved in order of registration by the USE LIBRARY statement.

You can override precedence or call functions from specific libraries by using the following syntax when you invoke the function:

*programid***!***functionname***(***argument1*,*argument2*,...**)**

The function name is preceded by the function library's program ID and an exclamation point (!). This syntax ensures that the correct function is called in cases where a function name is not unique among libraries.

If a library includes multiple interfaces, you can use the following syntax to specify the library and interface:

*programid***!***interfaceid***!***functionname***(***argument1*,*argument2*,...**)**

Analysis Services

How To

This section contains the administrative procedures for Microsoft® SQL Server™ 2000 Analysis Services. Procedure are listed in the following topics:

- Configuring Analysis Servers
- Creating Cubes' Prerequisite Objects
- Building and Processing Cubes
- Creating Security Roles
- Managing Cube Storage
- Enhancing Dimensions
- Enhancing Cubes with Optional Features
- Updating Cubes and Dimensions
- Managing Data Mining Models
- Archiving, Restoring, and Copying Data
- Analyzing and Optimizing Performance
- Automating and Scheduling Administrative Tasks

For more information about the background for a procedure, in the topic that lists the steps in the procedure, click a link in the See Also list.

See Also

Administering Analysis Services

Analysis Services

Configuring Analysis Servers

The following topics describe procedures for configuring Analysis servers:

- How to register an Analysis server
- How to add a linked Analysis server using SQL Server Enterprise Manager
- How to start the Migrate Repository Wizard
- How to configure Analysis Services for the Web

Analysis Services

How to register an Analysis server

To register a server

1. In the Analysis Manager tree pane, right-click the Analysis Servers folder, and then click **Register Server**.

2. Depending on the method of authentication, in the **Register Analysis Server** dialog box, type the following:

    - For a direct connection, which is the default authentication method, type the name of a server in the Microsoft® Windows NT® 4.0 or Microsoft Windows® 2000 system, and then click **OK**.

    - For a connection based on HTTP, type "http://" and the name or IP address of the Web server that hosts Analysis Services, and then click **OK**. In many cases, the name of the Web server will be the same as the name of the Analysis server. Establishing connections based on HTTP is possible only if you install Analysis Services for Microsoft SQL Server™ 2000 Enterprise Edition.

    Note You must have security authorization to access the Analysis server through both methods. Your user name must be included in the OLAP Administrators group on the Analysis server being registered, and if the Analysis server being registered is a remote computer, your user name must also be included in the OLAP Administrators group on the local computer. If your user name is added to the OLAP Administrators group, you may have to log off and log back on to Windows NT 4.0 or Windows 2000 before you can register the server.

See Also

Authentication of Connections

Authentication of Direct Connections

Registering Servers

Analysis Services

How to add a linked Analysis server using SQL Server Enterprise Manager

To add a linked Analysis server using SQL Server Enterprise Manager

1. In SQL Server Enterprise Manager, connect to the SQL Server computer that will host the link.

2. In the tree pane, expand the Security folder to view the Linked Servers.

3. Right-click Linked Servers, and then click **New Linked Server**.

4. In the **Linked Server Properties** dialog box, type information about the linked server. The required fields are:

    - **Linked Server**: This is the name of the Analysis server as it is referenced in SQL Server. This name is used to identify the linked server in queries.

    - **Provider Name**: Select **Microsoft OLE DB Provider for Olap Services 8.0**.

    - **Data Source**: This is the host name of the computer hosting Analysis Services.

    - **Catalog**: This is the name of the OLAP database provided through this link.

5. Click **Provider Options**. In the **Provider Options** dialog box, select the **Allow InProcess** check box.

    Note The **Allow InProcess** option affects all linked servers that use the OLE DB for Analysis Services provider. This option can be changed only in SQL Server Enterprise Manager.

See Also

Adding a Linked Server

Analysis Services

How to start the Migrate Repository Wizard

To start the Migrate Repository Wizard

1. In the Analysis Manager tree pane, expand the Analysis Servers folder.

2. Right-click the server whose Analysis Services repository you want to migrate, and then click **Migrate Repository**.

See Also

Migrating Analysis Services Repositories

Analysis Services

How to configure Analysis Services for the Web

To configure Analysis Services for the Web

1. Copy the Msolap.asp file from the C:\Program Files\Microsoft Analysis Services\Bin folder to the C:\Inetpub\Wwwroot folder of the computer that you want to use as an Analysis server. (The actual locations might differ from these default locations.) This computer must be accessible through a connection authenticated by Microsoft® Internet Information Services (IIS).

   IMPORTANT If the C:\Inetpub\Wwwroot folder does not exist on the Analysis server computer, IIS might not be installed. IIS must be installed on the Analysis server computer, and a name or IP address must be assigned to the default Web site before you can complete this procedure. The Analysis server computer must be running an operating system that supports IIS, such as Microsoft Windows NT® 4.0 Server (not Windows NT 4.0 Workstation) or Microsoft Windows® 2000.

2. Create or open a Microsoft Management Console (MMC) console containing the Internet Information Services snap-in.

   On Windows 2000, you can open Administrative Tools and run Internet Services Manager.

3. In the console tree, expand the server you are administering, and then click **Default Web Site**.

   If the server you want to administer does not appear in the console tree, right-click **Internet Information Services**, and then click **Connect** to specify the server.

4. In the details pane, right-click Msolap.asp, and then click **Properties**.

5. In the **Properties** dialog box, select the **Read** check box if it is not selected, and then select the **Script source access** check box. Click **OK** to set these properties and close the dialog box.

6. In the tree pane of the Internet Information Services snap-in, right-click the Web site, and then click **Properties**.

7. In the Web site properties dialog box, click the Home **Directory tab**. Depending on the operating system of the Analysis server computer, make the following changes for the site where Msolap.asp is hosted:
   - For Windows 2000, in the **Execute Permissions** box, select either **Scripts only** or **Scripts and Executables**. Optionally, in the **Application Protection** box, select **High (Isolated)** to ensure maximum stability for Analysis Services running as an IIS-based process.

   - For Windows NT 4.0 Server, click either **Script** or **Execute (including script)**. Optionally, select the **Run in separate memory space (isolated process)** check box to ensure maximum stability for Analysis Services running as an IIS-based process.
8. Click **OK**.

See Also

Authentication of Connections

How to register an Analysis server

Analysis Services

Creating Cubes' Prerequisite Objects

The following topics describe procedures for creating cubes' prerequisite objects:

- How to create a database
- How to specify a data source
- How to to create a shared dimension using the Dimension Wizard
- How to create a shared dimension using Dimension Editor
- How to create a virtual dimension based on member properties
- How to create a virtual dimension based on columns

See Also

Creating Prerequisite Objects For Cubes

Analysis Services

How to create a database

To create a database

1. In the Analysis Manager tree pane, under the Analysis Servers folder, right-click the server that will contain the database, and then click **New Database**.

2. In the **Database** dialog box, type a database name and description of your choice, and then click **OK**.

See Also

Creating Databases

Analysis Services

How to specify a data source

To specify a data source

1. In the Analysis Manager tree pane, expand the database that will draw data from the data source.

2. Right-click the Data Sources folder, and then click **New Data Source**.

3. In the **Data Link Properties** dialog box, on the **Provider** tab, click an OLE DB provider.

4. Complete the **Connection** tab, which varies by provider.

   Note If you are connecting to a SQL Server™ 2000 data provider, it is recommended that, under **Enter information to log on to the server**, you select **Use Windows NT Integrated security**.

   If, in Step 3 you selected Microsoft® OLE DB Provider for ODBC Drivers, on the **Connection** tab click **Use data source name**, click the drop-down arrow to display a list of data source names, and then click a data source name. If the data source name you want to select is not in the list, you must define one. In Control Panel, use the ODBC Data Source Administrator to define the data source name, and then return to the **Connection** tab and click **Refresh**. For more information, see Before Administering Analysis Services.

5. To determine whether the data source is correctly connected, click **Test Connection**. If the connection is correct, you receive a message indicating that the test connection succeeded. Click **OK** to clear the message.

6. In the **Data Link Properties** dialog box, click **OK**.

See Also

Specifying Data Sources

Analysis Services

How to create a shared dimension using the Dimension Wizard

To create a shared dimension using the Dimension Wizard

1. In the Analysis Manager tree pane, expand the database in which you want to create a shared dimension.

2. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Wizard**.

3. In the second step of the wizard select either **Star Schema: A single dimension table** or **Snowflake Schema: Multiple**, **related dimension tables**. The other options are for parent-child, virtual, and data mining dimensions.

4. Follow the remaining wizard steps.

After you complete the wizard, Dimension Editor appears so that you can further refine the dimension.

See Also

Creating a Shared Dimension with the Wizard

Analysis Services

How to create a shared dimension using Dimension Editor

To create a shared dimension using Dimension Editor

1. In the Analysis Manager tree pane, expand the database in which you want to create the shared dimension.

2. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Editor**.

3. In the **Choose a Dimension Table** dialog box, expand a data source, click the dimension table for the dimension, and then click **OK**.

   Dimension Editor appears with the dimension table showing in the **Schema** tab.

4. To add more tables to the dimension, on the **Insert** menu, click **Tables**, and then use the **Select Table** dialog box.

5. If the dimension contains multiple tables, ensure each is joined to another. To join two columns, in the **Schema** tab, drag one column to the other.

6. Create levels in the dimension. It is easiest to create the highest, most general level first and then create progressively lower, more specific levels; otherwise, if member counting is enabled, a confirmation dialog box appears each time you add a level. For each level you want to create:

   a. On the **Insert** menu, click **Level**.

   b. In the **Insert Level** dialog box, select the column on which the level is based, and then click **OK**.

As you create each level, it appears in the tree pane.

7. (Optional.) In the properties pane, modify the properties of the dimension and its levels. For information about these properties, see Properties Pane (Dimension Editor Data View).

8. To save the dimension, on the **File** menu, click **Save**, type a name for the dimension in the **New Dimension Name** dialog box, and then click **OK**.

See Also

Creating a Shared Dimension with the Editor

Analysis Services

How to browse a shared dimension

To browse a shared dimension

- In the Analysis Manager tree pane, right-click a shared dimension, and then click **Browse Dimension Data**.

  -or-

- In the Cube Editor tree pane, right-click a shared dimension, and then click **Browse**.

See Also

Viewing Dimension Data

Analysis Services

How to browse a private dimension

To browse a private dimension

- In the Cube Editor tree pane, right-click a private dimension, and then click **Browse**.

See Also

Viewing Dimension Data

Analysis Services

How to create a virtual dimension based on member properties

To create a virtual dimension based on member properties

1. Depending on whether the virtual dimension will be shared or private, do one of the following:
    - If the virtual dimension will be shared, in the Analysis Manager tree pane, expand the database in which you want to create the virtual dimension. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Wizard**.
    - If the virtual dimension will be private, in the Analysis Manager tree pane, right-click the cube in which you want to create the virtual dimension, and then click **Edit**. On the **Insert** menu, point to **Dimension**, and then click **New**.
2. In the **Welcome** step, click **Next**.
3. In the **Choose how you want to create the dimension** step, select **Virtual Dimension: The member properties of another dimension**. Click **Next**.
4. In the **Select the dimension with the member properties** step, select the dimension that contains the member properties. Click **Next**.
5. In the **Select the levels for the virtual dimension** step, select the levels for the virtual dimension:
    a. For each level you want to define, beginning with the top level, in the **Available member properties** box, select the member property that defines the level, and then click >.

    b. Click **Next**.

6. In the **Select advanced options** step, choose any advanced options that may apply. If no advanced options apply, click **Next**.

7. If the **Specify ordering and uniqueness** step appears, for each level, in the **Order by** column, select the value that determines the order by which the level's members are displayed to end users. Also, for each level, in the **Keys unique** and **Names unique** columns, specify the scope of uniqueness among the member keys and member names, respectively. Click **Next**.

8. In the **Finish** step:
    a. In the **Dimension name** box, type a name up to 24 characters long.

    b. If the virtual dimension will be private, clear the **Share this dimension with other cubes** check box. (This check box is not displayed if you are following the procedure for a shared virtual dimension.)

    c. Click **Finish**.

See Also

Creating Virtual Dimensions

Analysis Services

How to create a virtual dimension based on columns

To create a virtual dimension based on columns

1. Depending on whether the virtual dimension will be shared or private, do one of the following:
    - If the virtual dimension will be shared, in the Analysis Manager tree pane, expand the database in which you want to create the virtual dimension. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Wizard**.
    - If the virtual dimension will be private, in the Analysis Manager tree pane, right-click the cube in which you want to create the virtual dimension, and then click **Edit**. On the **Insert** menu, point to **Dimension**, and then click **New**.
2. In the **Welcome** step, click **Next**.
3. In the **Choose how you want to create the dimension** step, do one of the following:
    - If the dimension that will supply the columns has a single dimension table, select **Star Schema: A single dimension table)**.
    - If the dimension that will supply the columns has multiple dimension tables, select **Snowflake Schema: Multiple, related dimension tables**.
4. Click **Next**.
5. In the **Select the dimension table(s)** step, select the dimension table or tables that will supply the columns for the virtual dimension. Click **Next**.

6. If the **Create and edit joins** step appears, join all tables by dragging columns to their joining columns. Click **Next**.

7. If the **Select the dimension type** step appears, select the dimension type. If you select **Time dimension**:

    a. In the **Date column** box, select the column that will be parsed to create the levels of the virtual dimension.

    b. Click **Next**.

    c. In the **Select time levels** box, select the levels for the virtual dimension.

    d. (Optional.) In the **Day** and **Month** boxes, select the starting date of the year.

    e. Click **Next**.

    If you select **Standard dimension**, click **Next**.

8. If the **Select the levels for your dimension** step appears, select the levels for the virtual dimension:

    a. For each level you want to define, beginning with the top level, in the **Available columns** box, select the column that defines the level, and then click **>**.

    b. Click **Next**.

9. If the **Specify the member key columns** step appears, select member key columns for one or more of the levels. Click **Next**.

10. In the **Select advanced options** step, choose any advanced options

that may apply, and then click **Next**. Depending on the advanced options you select, other dialog boxes may appear.

11. In the **Set Changing property** step, select **Changing**. A virtual dimension must be a changing dimension. Click **Next**.

12. If the **Specify ordering and uniqueness** step appears, for each level, in the Order by column, select the value that determines the order by which the level's members are displayed to end users. Also, for each level, in the **Keys unique** and **Names unique** columns, specify the scope of uniqueness among the member keys and member names, respectively. Click **Next**.

13. If the **Specify storage mode and create member groups** step appears, specify a storage mode. If you select **MOLAP**, indicate whether to create member groups in a level inserted automatically above the dimension's lowest level. Click **Next**.

14. In the **Finish** step:

    a. In the **Dimension name** box, type a name up to 24 characters long.

    b. If the virtual dimension will be private, clear the **Share this dimension with other cubes** check box. (This check box is not displayed if you are following the procedure for a shared virtual dimension.)

    c. Click **Finish**.

15. In the editor tree pane, select the newly created virtual dimension.

16. If the properties pane is not expanded, expand it by clicking **Properties** beneath the tree pane.

17. In the properties pane, click the **Advanced** tab.

18. Click the **Depends on Dimension** property, and in the drop-down list click the dimension that supplies the columns for the virtual dimension.

19. Click the **Virtual** property, and in the drop-down list click **True**.

20. On the **File** menu, click **Save**.

See Also

Creating Virtual Dimensions

Analysis Services

Building and Processing Cubes

The following topics describe procedures for building and processing cubes:

- How to start the Cube Wizard
- How to build a cube with Cube Editor
- How to create a private dimension
- How to add a multiple-column measure to a cube
- How to start the Virtual Cube Wizard
- How to start the Storage Design Wizard
- How to process a cube
- How to process a virtual cube
- How to view an SQL statement
- How to change the temporary file folder used by Analysis_Services
- How to browse sample data in Cube Editor

See Also

Building and Processing Cubes

Analysis Services

How to start the Cube Wizard

To start the Cube Wizard

1. In the Analysis Manager tree pane, expand the database in which you want to build a cube.

2. Right-click the Cubes folder, point to **New Cube**, and then click **Wizard**.

After you complete the wizard, Cube Editor appears so that you can further refine your cube.

See Also

Building a Cube with the Wizard

Cubes

Analysis Services

How to build a cube with Cube Editor

To build a cube with Cube Editor

1. In the Analysis Manager tree pane, expand the database in which you want to build a cube.

2. Right-click the Cubes folder, point to **New Cube**, and then click **Editor**.

3. In the **Choose a fact table** dialog box, expand a data source, and then click the table to use as your cube's fact table.

   The columns of the table you click appear under **Details**.

4. Click **OK**.

   Cube Editor appears, with your fact table showing in the **Schema** tab.

5. To select measures for your cube, in the Cube Editor tree pane, right-click the Measures folder, and then click **New Measure**. Use the **Insert Measure** dialog box to select columns in your fact table as measures.

   As you select each measure, it appears in the tree pane under the Measures folder.

6. Select existing shared dimensions or define new dimensions for your cube.

   - Existing shared dimensions: In the Cube Editor tree pane, right-click the Dimensions folder, and then click **Existing Dimensions**. Select existing shared dimensions under **Shared dimensions**, and then click > to add them to your cube.

   - New dimensions: In the Cube Editor tree pane, right-click the Dimensions folder, and then click **New Dimension**. Use the Dimension Wizard to create a new dimension. If you want the dimension to be private, in the **Finish** step, clear the **Share this dimension with other cubes** check box.

After you select or define a dimension, its dimension table appears in the **Schema** tab of Cube Editor, with the join between the dimension table and the fact table illustrated by a line. Also, a dimension table can be joined to another dimension table to form a snowflake schema. Verify that all desired joins are present. To add a join, drag a column to the joining column.

7. To save your cube, on the **File** menu, click **Save**, type a name for your cube in the **New Cube Name** dialog box, and then click **OK**.

   The new name for your cube appears in the Cube Editor tree pane.

8. To design the aggregations for your cube, on the **Tools** menu, click **Design Storage**. The Storage Design Wizard appears to help you design aggregations.

   The last step of the wizard gives you the option to process your cube now or later. You must process it before you and end users can view its data. Depending on the size of your cube, processing can take considerable time.

Note Newly processed cubes are visible to end users only after they reconnect to the server computer.

See Also

Building a Cube with the Editor

Cubes

Designing Storage Options and Aggregations

Processing Cubes

Analysis Services

How to create a private dimension

To create a private dimension

1. In the Analysis Manager tree pane, right-click the cube in which you want to create a private dimension, and then click **Edit**.

2. In Cube Editor, on the **Insert** menu, point to **Dimension**, and then click **New**.

3. In the Dimension Wizard, advance through the steps to define the private dimension. For more information, in the wizard, click **Help**.

4. In the **Finish** step, to indicate the dimension is private, clear the **Share this dimension with other cubes** check box.

5. Click **Finish**.

6. (Optional.) In the properties pane, modify the properties of the dimension and its levels.

7. On the **File** menu, click **Save**.

See Also

Creating and Maintaining Private Dimensions

Properties Pane (Dimension Editor Data View)

Analysis Services

How to add a multiple-column measure to a cube

To add a multiple-column measure to a cube

1. In the Analysis Manager tree pane, right-click the cube, and then click **Edit**.

2. In the Cube Editor tree pane, right-click the Measures folder, and then click **New Measure**. Use the **Insert Measure** dialog box to select one of the columns for the measure.

3. In the Cube Editor tree pane, ensure the new measure is selected.

4. Expand the properties pane, and in the **Source Column** value, type an expression containing the columns. For example:
   ```
   "sales_fact_1998"."store_sales"-"sales_fact_1998"."store_cos
   ```

5. On the **File** menu, click **Save**.

See Also

Adding a Multiple-Column Measure to a Cube

Analysis Services

How to start the Virtual Cube Wizard

To start the Virtual Cube Wizard

1. In the Analysis Manager tree pane, expand the database in which you want to create a virtual cube.

2. Right-click the Cubes folder, and then click **New Virtual Cube**.

See Also

Building a Virtual Cube

Analysis Services

How to start the Storage Design Wizard

To start the Storage Design Wizard

1. In the Analysis Manager tree pane, under the database that contains the cube for which you want to set storage options and design aggregations, expand the Cubes folder.

2. Right-click the cube for which you want to set storage options and design aggregations, and then click **Design Storage**.

See Also

Designing Storage Options and Aggregations

Analysis Services

How to process a cube

To process a cube

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, and then click **Process**.

3. In the **Process a Cube** dialog box, click **Full Process**, and then click **OK**.

4. In the **Process** dialog box, wait for the cube to finish processing, or click **Stop** to halt and cancel processing.

After processing completes, but before you close the **Process** dialog box, you can view the SQL statement used to process the cube.

Note Newly processed cubes are visible to end users only after they reconnect to the server computer.

See Also

How to view an SQL statement

Processing Cubes

Analysis Services

How to process a virtual cube

To process a virtual cube

1. In the Analysis Manager tree pane, under the database that contains the virtual cube, expand the Cubes folder.

2. Right-click the virtual cube, and then click **Process**.

3. In the **Process** dialog box, wait for the virtual cube to finish processing, or click **Stop** to halt and cancel processing.

After processing completes, but before you close the **Process** dialog box, you can view the SQL statement used to process the virtual cube.

Note Newly processed virtual cubes are visible to end users only after they reconnect to the server computer.

See Also

How to view an SQL statement

Processing Cubes

Analysis Services

How to view an SQL statement

To view an SQL statement

1. In the **Process** dialog box, click a line beginning with the SQL icon.

2. Click **View Details**.

See Also

Creating Partitions

Processing Cubes

Updating and Refreshing Cube Data

Analysis Services

How to change the temporary file folder used by Analysis Services

To change the temporary file folder used by Analysis Services

1. In the Analysis Manager tree pane, right-click the Analysis server for which you want to change the temporary file folder, and then click **Properties**.

2. Beside the **Temporary file folder** box, click **Browse**, select a new folder, and then click **OK**.

3. In the **Properties** dialog box, click **OK**.

If you change the folder, you must stop and restart the MSSQLServerOLAPService service.

See Also

Processing Cubes

Analysis Services

How to browse sample data in Cube Editor

To browse sample data in Cube Editor

1. In the Analysis Manager tree pane, under the Cubes folder, right-click an unprocessed cube, and then click **Edit**.

2. In Cube Editor, on the **View** menu, click **Data**.

Analysis Services displays the **Data** tab of Cube Editor with sample data in the data viewing pane. The following message is displayed at the bottom of Cube Editor:

! Cube is not processed. Viewing sample data.

See Also

Browsing an Unprocessed Cube

Analysis Services

Creating Security Roles

The following topics describe procedures for creating security roles:

- How to create a database role
- How to create a cube role, change its default values, and specify cell security
- How to create a mining model role and change its default values
- How to create a custom rule for dimension security in a database role
- How to create a custom rule for dimension security in a cube role
- How to create a custom rule for cell security

See Also

Creating Security Roles

Analysis Services

How to create a database role

To create a database role

1. In the Analysis Manager tree pane, right-click the database for which you want to create a database role, and then click **Manage Roles**.

2. In Database Role Manager, do one of the following:
   - To use an existing role as the basis for the new role, select the existing role, and then click **Duplicate**. In the **Duplicate Role** dialog box, specify a name for the new role, and then click **OK**. Select the new role, and then click **Edit**.

   - To define the new role without values from another role, click **New**. In the **Database Role** dialog box, type a value in the **Role name** box. You can enter a maximum of 50 characters; the name must begin with an alphabetical character.
3. (Optional.) In the **Database Role** dialog box, type a value in the **Description** box.

4. In the **Enforce on** box, select one of the following:
   - **Server**. Server enforcement is more secure due to filtering of data on the server, but this may slow performance. Queries are resolved entirely on the Analysis server or at the data source.

   - **Client**. Client enforcement generally provides better performance but may allow users to gain unauthorized access to data on the client workstation. Queries might be resolved partially or completely at the client workstation.
5. In the **Membership** tab, specify the users and groups in the role. To begin adding users and groups, click **Add**, and then in the **Add Users and Groups** dialog box:

a. In the **List Names From** list, click the domain from which to select users and groups.

b. To display users under **Names**, click **Show Users**.

c. To display a group's members, click the group, and then click **Members**.

d. To add a user or group to the role, click the user or group, and then click **Add**.

e. After you finish adding the users and groups to the role, click **OK**.

To remove a user or group from the role, in the **Membership** tab, select the user or group, and then click **Remove**.

6. (Optional.) In the **Cubes** tab, select the cubes that the database role can access. For each cube you select, a cube role is created.

7. (Optional.) In the **Mining Models** tab, select the data mining models that the database role can access. For each mining model you select, a mining model role is created.

8. (Optional.) In the **Dimensions** tab, for each displayed permission, select a rule. (A read/write permission appears only for a write-enabled dimension.) The following table describes the rules that are available for each permission.

Permission	Rule	Rule description
Read	Unrestricted	The role can view all members. This rule is the default.
	Fully Restricted	The role can view only a single member. If the dimension does not have an (All)

		level, then the visible member is the first member in the topmost level.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be viewed. To access this dialog box, select **Custom**, and then in the **Custom Settings** column, click the edit (**...**) button. For more information, see Defining Custom Rules for Dimension Security.
Read/write	Unrestricted	The role can update all members. This rule is available only if the rule for the read permission is Unrestricted.
	Fully Restricted	The role cannot update members. This rule is the default and is available only if the rule for the read permission is Unrestricted or Fully Restricted.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be updated. To access this dialog box, select **Custom**, and then in the **Custom Settings** column click the edit (**...**) button. This rule is available only if the rule for the read permission is Unrestricted or Custom. For more information, see Defining Custom Rules for Dimension Security.

For more information about these permissions and rules, see Dimension Security.

9. In the **Database Role** dialog box, click **OK**.

The read/write permission in the **Dimensions** tab is effective only as long as the dimension remains write-enabled. For more information, see Write-Enabled Dimensions.

See Also

Creating Database Roles

Analysis Services

How to create a cube role, change its default values, and specify cell security

To create a cube role, change its default values, and specify cell security

1. In the Analysis Manager tree pane, right-click the cube for which you want to create a cube role, and then click **Manage Roles**.

2. In Cube Role Manager, unchecked roles are database roles without access to the cube. Checked roles are cube roles, which have access. Do one of the following:
   - To create a new cube role by granting access to a database role, select the check box beside the database role. The remaining steps in this procedure are optional. To continue, select the cube role, and then click **Edit**.

   - To use an existing cube role as the basis for the new cube role, select the existing cube role, and then click **Duplicate**. (This action also creates a database role with the same name as the new cube role.) In the **Duplicate Role** dialog box, specify a name for the new role that is 50 characters or less and begins with an alphabetical character, and then click **OK**. Select the new role, and then click **Edit**.

   - To define the new cube role without values from another role, click **New**. (This action also creates a database role with the same name as the cube role.) In the **Cube Role** dialog box, type a value in the **Role name** box. You can enter a maximum of 50 characters; the name must begin with an alphabetical character.
3. (Optional.) In the **Cube Role** dialog box, type a value in the **Description** box.

4. In the **Enforce on** box, select one of the following:
    - **Server**. Server enforcement is more secure but may slow performance. Queries are resolved entirely on the Analysis server or at the data source.
    - **Client**. Client enforcement generally provides better performance but may allow users to gain unauthorized access to data on the client workstation. Queries might be resolved partially or completely at the client workstation.
5. In the **Enable drillthrough** check box, indicate whether the role can drill through to the source data for a cell. This ability also requires that you enable drillthrough for the cube or at least one of its partitions. For more information, see Specifying Drillthrough Options.
6. In the **Membership** tab, specify the users and groups in the role.

    Note Changes in this tab propagate to the database role and cube roles with the same name as the edited cube role.

    To begin adding users and groups, click **Add**, and then in the **Add Users and Groups** dialog box:

    a. In the **List Names From** list, click the domain from which to select users and groups.

    b. To display users under **Names**, click **Show Users**.

    c. To display a group's members, click the group, and then click **Members**.

    d. To add a user or group to the role, click the user or group, and then click **Add**.

    e. After you have added the users and groups to the role, click

OK.

To remove a user or group from the role, in the **Membership** tab, select the user or group, and then click **Remove**.

7. (Optional.) In the **Dimensions** tab, for each displayed permission, select a rule. (A read/write permission appears only for a write-enabled dimension.) The following table describes the rules that are available for each permission.

Permission	Rule	Rule description
Read	Unrestricted	The role can view all members. This rule is the default.
	Fully Restricted	The role cannot view members. When users in the role browse the cube, they do not see the dimension.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be viewed. To access this dialog box, select **Custom**, and then in the **Custom Settings** column, click the edit (**...**) button. For more information, see Defining Custom Rules for Dimension Security.
Read/write	Unrestricted	The role can update all members. This rule is available only if the rule for the read permission is Unrestricted.
	Fully Restricted	The role cannot update members. This rule is the default and is available only if the rule for the read permission is Unrestricted or Fully Restricted.
	Custom	Only the levels and members you specify in the **Custom Dimension Security** dialog box can be updated. To access this dialog box, select **Custom**, and then in the **Custom Settings** column, click the edit (**...**) button. This rule is available

		only if the rule for the read permission is Unrestricted or Custom. For more information, see Defining Custom Rules for Dimension Security.

Changes to a read/write permission propagate to the database role of the same name. For more information about these permissions and rules, see Dimension Security.

8. (Optional.) In the **Cells** tab, in the **Cell security policy** box, select one of the following three policies:
    - **Unrestricted read**

      The role can view all cell values. This policy is the default.
    - **Unrestricted read/write**

      The role can view and update all cell values. This policy is available only if the cube you selected in Step 1 is write-enabled or if the virtual cube you selected in Step 1 has one or more write-enabled, component cubes.
    - **Advanced**

      The role can view and update only the cell values you specify in the permissions and rules in the **Cells** tab.
    - **Allow users to commit writeback changes**

      This option is available only for write-enabled cubes with an Advanced cell security policy. If this option is selected, changes are permanently recorded in the writeback table. If this option is not selected, changes apply only to ad hoc analysis and are temporary.
9. (Optional.) If in the preceding step you selected the Advanced policy, select a rule for each permission displayed in the **Cells** tab. (A read/write permission appears only if in Step 1 the cube you selected is write-enabled, or if the virtual cube you selected has one or more write-enabled, component cubes.) The following table describes the

rules that are available for each permission.

Permission	Rule	Rule description
Read	Unrestricted	The role can view all cell values. This rule is the default.
	Fully Restricted	The role can view only the cell values specified in the read/write permission or read contingent permission, subject to its limitations. For more information about the limitations of the read contingent permission, see Cell Security.
	Custom	You can specify the cell values that are viewable and not viewable in the **Cube Cell Security** dialog box. To access this dialog box, select **Custom**, and then in the **Custom Settings** column, click the edit (**...**) button.
Read contingent	Unrestricted	The role can view all cell values that are not derived from other cells. If a cell value is derived from other cells, it is viewable if all the other cells are included in the read or read/write permission.
	Fully Restricted	The role can view only the cell values specified in the read permission or read/write permission. This rule is the default.
	Custom	You can specify the cell values that are viewable and not viewable, subject to the limitations of the read contingent permission. (For more information about the limitations of the read contingent permission, see Cell Security.) To do this, use the **Cube Cell Security** dialog box. To access this dialog box, select **Custom**, and then in the **Custom**

		Settings column, click the edit (**...**) button.
Read/write	Unrestricted	The role can update all cell values.
	Fully Restricted	The role cannot update cell values.
	Custom	You can specify the cell values that are updatable and not updatable in the **Cube Cell Security** dialog box. To access this dialog box, select **Custom**, and then in the **Custom Settings** column, click the edit (**...**) button.

For more information about these permissions and rules, see Cell Security.

10. In the **Cube Role** dialog box, click **OK**.

The read/write permission in the **Dimensions** tab is effective only as long as the dimension remains write-enabled. For more information, see Write-Enabled Dimensions.

The Unrestricted Read/Write policy and read/write permission in the **Cells** tab are effective only as long as the cube remains write-enabled. For more information, see Maintaining Write-Enabled Cubes and Writeback Data.

See Also

Creating Cube Roles

Analysis Services

How to create a mining model role and change its default values

To create a mining model role and change its default values

1. In the Analysis Manager tree pane, right-click the mining model for which you want to create a mining model role, and then click **Manage Roles**.

2. In Mining Model Role Manager, unchecked roles are database roles without access to the mining model. Checked roles are mining model roles, which have access. Do one of the following:

    - To create a new mining model role by granting access to a database role, select the check box beside the database role. The remaining steps in this procedure are optional. To continue, select the mining model role, and then click **Edit**.

    - To use an existing mining model role as the basis for the new mining model role, select the existing mining model role, and then click **Duplicate**. (This action also creates a database role with the same name as the new mining model role.) In the **Duplicate Role** dialog box, specify a name for the new role that is 50 characters or less and begins with an alphabetical character, and then click **OK**. Select the new role, and then click **Edit**.

    - To define the new mining model role without values from another role, click **New**. (This action also creates a database role with the same name as the mining model role.) In the **Mining Model Role** dialog box, type a value in the **Role name** box. You can enter a maximum of 50 characters; the name must begin with an alphabetical character.

3. (Optional.) In the **Mining Model Role** dialog box, type a value in the

Description box.

4. In the **Membership** tab, specify the users and groups in the role.

   Note Changes in this tab propagate to the database role and mining model roles with the same name as the edited mining model role.

   To begin adding users and groups, click **Add**, and then in the **Add Users and Groups** dialog box:

   a. In the **List Names From** list, click the domain from which to select users and groups.

   b. To display users under **Names**, click **Show Users**.

   c. To display a group's members, click the group, and then click **Members**.

   d. To add a user or group to the role, click the user or group, and then click **Add**.

   e. After you have added the users and groups to the role, click **OK**.

   To remove a user or group from the role, in the **Membership** tab, select the user or group, and then click **Remove**.

5. In the **Mining Model Role** dialog box, click **OK**.

See Also

Creating Mining Model Roles

Analysis Services

How to create a custom rule for dimension security in a database role

To create a custom rule for dimension security in a database role

1. In the Analysis Manager tree pane, right-click the database that contains the database role, and then click **Manage Roles**.

2. In Database Role Manager, select the database role in which you want to create a custom rule, and then click **Edit**. If the database role does not yet exist, you must create it before performing this procedure. For more information, see Creating Database Roles.

3. In the **Database Role** dialog box, click the **Dimensions** tab.

4. In the row displaying the dimension and permission for which you want to create a custom rule, in the **Rule** column, select **Custom**. For more information about permissions, see Dimension Security.

5. In the same row, in the **Custom Settings** column, click the edit (**...**) button.

6. (Optional.) In the **Custom Dimension Security** dialog box, in the **Description** box, type a description of the custom rule.

7. If you want to limit access to levels or members, decide whether to use the **Basic** tab or **Advanced** tab, which provide alternative methods. The **Basic** tab is easier to use and satisfies most needs. The **Advanced** tab provides a little more flexibility, but you must write Multidimensional Expressions (MDX).

8. (Optional.) To use the **Basic** tab to limit access to levels or members:

    a. Click the **Basic** tab.

    b. (Optional.) In the **Top Level** box, select the topmost accessible level.

    c. (Optional.) In the **Bottom Level** box, select the bottommost accessible level.

    d. Select **Select all members** to select all members in the **Members** box, or select **Deselect all members** to clear the selection of all members in the **Members** box.

    e. (Optional.) In the **Members** box, select a check box beside a member to allow access to it. (This action also selects the member's descendants and ancestors that are visible in the **Members** box.) Clear a check box beside a member to deny access to it. (This action also clears the selection of the member's descendants that are visible in the **Members** box.)

       Important If you limit access to levels or members, consider selecting **Enable - Show visual totals** in the **Common** tab. If instead you use the default setting for visual totals (that is, **Disable - Do not show visual totals**), security exposures might be created. These exposures allow end users in the role to deduce values for members to which they are denied access. For more information, see Example 4, Example 6, Example 8, and Example 9 in Custom Rules in Dimension Security.

9. (Optional.) To use the **Advanced** tab to limit access to levels or members, click the **Advanced** tab, and use one or more of the following boxes:

    - In the **Top Level** box, type an MDX expression that evaluates to a level that will be the topmost accessible level.

- In the **Bottom Level** box, type an MDX expression that will evaluate to a level that will be the bottommost accessible level.

- In the **Allowed Members** box, type an MDX expression for the set of members that can be accessed. Descendants of these members can also be accessed unless they are below the bottom level or access to them is denied by the expression in the **Denied Members** box. The ancestors of the allowed members will be visible at the top level.

- In the **Denied Members** box, type an MDX expression for the set of members that cannot be accessed. Descendants of these members cannot be accessed unless access to them is allowed by the expression in the **Allowed Members** box.

For more information, see Custom Rules in Dimension Security.

IMPORTANT If you limit access to levels or members, consider selecting **Enable - Show visual totals** in the **Common** tab. If instead you use the default setting for visual totals (that is, **Disable - Do not show visual totals**), security exposures might be created. These exposures allow end users in the role to deduce values for members to which they are denied access. For more information, see Example 4, Example 6, Example 8, and Example 9 in Custom Rules in Dimension Security.

10. (Optional.) To control visual totals, click the **Common** tab, and in the **Visual Totals** area, do one of the following:

    - To enable visual totals at all viewable levels, select **Enable - Show visual totals**.

    - To disable visual totals at all viewable levels, select **Disable - Do not show visual totals**.

- To enable visual totals at and above a certain level but disable them below it, select **Custom - Show visual totals starting at the following level and above**, and in the box below it type an MDX expression for the level. Or, instead of typing, beside the box you can click the edit (**...**) button to access MDX Builder, where you can select the level in the **Data** box and drag it to the **MDX expression** box.

Note Visual totals cannot be enabled for a cube that contains a distinct count measure. For more information, see Using Aggregate Functions.

11. (Optional.) To select a default member, click the **Common** tab, select **Define default member and specify using MDX**, and in the box below it type an MDX expression for the default member. Or, instead of typing, beside the box you can click the edit (**...**) button to access MDX Builder, where you can select the default member in the **Data** box and drag it to the **MDX expression** box.

12. Click **OK**.

13. In the **Database Role** dialog box, click **OK**.

See Also

Defining Custom Rules for Dimension Security

Analysis Services

How to create a custom rule for dimension security in a cube role

To create a custom rule for dimension security in a cube role

1. In the Analysis Manager tree pane, right-click the cube associated with the cube role in which you want to create a custom rule, and then click **Manage Roles**.

2. In Cube Role Manager, select the cube role in which you want to create a custom rule, and then click **Edit**. If the cube role does not yet exist, you must create it before performing this procedure. For more information, see Creating Cube Roles.

3. In the **Cube Role** dialog box, click the **Dimensions** tab.

4. In the same row, in the **Custom Settings** column, click the edit (**...**) button.

5. (Optional.) In the **Custom Dimension Security** dialog box, in the **Description** box, type a description of the custom rule.

6. In the row displaying the dimension and permission for which you want to create a custom rule, in the **Rule** column, select **Custom**. For information about permissions, see Dimension Security.

7. If you want to limit access to levels or members, decide whether to use the **Basic** tab or **Advanced** tab, which provide alternative methods. The **Basic** tab is easier to use and satisfies most needs. The **Advanced** tab provides a little more flexibility, but you must write Multidimensional Expressions (MDX) expressions.

8. (Optional.) To use the **Basic** tab to limit access to levels or members:
    a. Click the **Basic** tab.

    b. (Optional.) In the **Top Level** box, select the topmost accessible level.

    c. (Optional.) In the **Bottom Level** box, select the bottommost accessible level.

    d. Select **Select all members** to select all members in the **Members** box, or select **Deselect all members** to clear the selection of all members in the **Members** box.

    e. (Optional.) In the **Members** box, select a check box beside a member to allow access to it. (This action also selects the member's descendants and ancestors that are visible in the **Members** box.) Clear a check box beside a member to deny access to it. (This action also clears the selection of the member's descendants that are visible in the **Members** box.)

    IMPORTANT If you limit access to levels or members, consider selecting **Enable - Show visual totals** in the **Common** tab. If instead you use the default setting for visual totals (that is, **Disable - Do not show visual totals**), security exposures might be created. These exposures allow end users in the role to deduce values for members to which they are denied access. For more information, see Example 4, Example 6, Example 8, and Example 9 in Custom Rules in Dimension Security.

9. (Optional.) To use the **Advanced** tab to limit access to levels or members, click the **Advanced** tab, and use one or more of the following boxes:
    - In the **Top Level** box, type an MDX expression that evaluates

to a level that will be the topmost accessible level.

- In the **Bottom Level** box, type an MDX expression that evaluates to a level that will be the bottommost accessible level.

- In the **Allowed Members** box, type an MDX expression for the set of members that can be accessed. Descendants of these members can also be accessed unless they are below the bottom level or access to them is denied by the expression in the **Denied Members** box. The ancestors of the allowed members will be visible at the top level.

- In the **Denied Members** box, type an MDX expression for the set of members that cannot be accessed. Descendants of these members cannot be accessed unless access to them is allowed by the expression in the **Allowed Members** box.

For more information, see Custom Rules in Dimension Security.

IMPORTANT If you limit access to levels or members, consider selecting **Enable - Show visual totals** in the **Common** tab. If instead you use the default setting for visual totals (that is, **Disable - Do not show visual totals**), security exposures might be created. These exposures allow end users in the role to deduce values for members to which they are denied access. For more information, see Example 4, Example 6, Example 8, and Example 9 in Custom Rules in Dimension Security.

10. (Optional.) To control visual totals, click the **Common** tab, and in the **Visual Totals** area, do one of the following:

    - To enable visual totals at all viewable levels, select **Enable - Show visual totals**.

    - To disable visual totals at all viewable levels, select **Disable -**

Do not show visual totals.

- To enable visual totals at and above a certain level but disable them below it, select **Custom - Show visual totals starting at the following level and above**, and in the box below it type an MDX expression for the level. Or, instead of typing, beside the box you can click the edit (**...**) button to access MDX Builder, where you can select the level in the **Data** box and drag it to the **MDX expression** box.

Note Visual totals cannot be enabled for a cube that contains a distinct count measure. For more information, see Using Aggregate Functions.

11. (Optional.) To select a default member, click the **Common** tab, select **Define default member and specify using MDX**, and in the box below it type an MDX expression for the default member. Or, instead of typing, beside the box you can click the edit (**...**) button to access MDX Builder, where you can select the default member in the **Data** box and drag it to the **MDX expression** box.

12. Click **OK**.

13. In the **Cube Role** dialog box, click **OK**.

See Also

Defining Custom Rules for Dimension Security

Analysis Services

How to create a custom rule for cell security

To create a custom rule for cell security

1. In the Analysis Manager tree pane, right-click the cube associated with the cube role in which you want to create a custom rule, and then click **Manage Roles**.

2. In Cube Role Manager, select the cube role in which you want to create a custom rule, and then click **Edit**. If the cube role does not yet exist, you must create it before performing this procedure. For more information, see Creating Cube Roles.

3. In the **Cube Role** dialog box, click the **Cells** tab.

4. In the **Cell security policy** box, select **Advanced**.

5. In the row displaying the permission for which you want to create a custom rule, in the **Rule** column, select **Custom**. For information about permissions, see Cell Security.

6. In the same row, in the **Custom Settings** column, click the edit (**...**) button.

7. (Optional.) In the **Cube Cell Security** dialog box, in the **Description** box, type a description of the custom rule.

8. In the **MDX** box, type a Multidimensional Expressions (MDX) expression to allow or deny access to specific cube cells. Or, instead of typing, beside the box you can click the edit (**...**) button to access MDX Builder, where you can create the expression by using drag and drop techniques. For more information, see Custom Rules in Cell

Security.

9. Click **OK**.

10. In the **Cube Role** dialog box, click **OK**.

See Also

Defining Custom Rules for Cell Security

Analysis Services

Managing Cube Storage

The following topics describe procedures for managing cube storage:

- How to start the Partition Wizard
- How to process a partition
- How to merge two partitions
- How to specify a data source for a linked cube
- How to create a linked cube
- How to process a linked cube

See Also

Introduction to Partitions

Partition Wizard

Linked Cubes

Analysis Services

How to start the Partition Wizard

To start the Partition Wizard

1. In the Analysis Manager tree pane, under the database in which you want to create a partition, expand the Cubes folder, and then expand the cube in which you want to create a partition.

2. Right-click the Partitions folder, and then click **New Partition**.

See Also

Creating Partitions

Analysis Services

How to process a partition

To process a partition

1. In the Analysis Manager tree pane, under the database that contains the partition, expand the Cubes folder, expand the cube that contains the partition, and then expand the Partitions folder.

2. Right-click the partition, and then click **Process**.

3. In the **Process** dialog box, wait for the partition to finish processing, or click **Stop** to halt and cancel processing.

After processing completes but before you close the **Process** dialog box, you can view the SQL statement used to process the partition.

How to view an SQL statement

See Also

Creating Partitions

Analysis Services

How to merge two partitions

To merge two partitions

1. In the Analysis Manager tree pane, under the database that contains the partitions, expand the Cubes folder, expand the cube that contains the partitions, and then expand the Partitions folder.

2. Right-click the source partition, and then click **Merge**.

3. In the **Merge Partitions** dialog box, select the target partition, and then click **Merge**.

4. In the **Process** dialog box, wait for the merge to finish processing, or click **Stop** to halt and cancel processing.

See Also

Merging Partitions

Analysis Services

How to specify a data source for a linked cube

To specify a data source for a linked cube

1. In the Analysis Manager tree pane, expand the database in which you want to create a linked cube.

2. Right-click the Data Sources folder, and then click **New Data Source**.

3. In the **Data Link Properties** dialog box, on the **Provider** tab, click **Microsoft OLE DB Provider for Olap Services 8.0** or another provider that is compliant with the OLAP section of the OLE DB specification dated March 1999 (2.6).

4. On the **Connection** tab, in the **Data Source** box, type the name of the Analysis server where the source cube is defined.

5. In the **Enter the initial catalog to use** box, type the name of the database where the source cube is defined.

6. To determine whether the data source is correctly connected, click **Test Connection**. If the connection is correct, you receive a message indicating that the test connection succeeded. Click **OK** to clear the message.

7. In the **Data Link Properties** dialog box, click **OK**.

See Also

Linked Cubes

Analysis Services

How to create a linked cube

To create a linked cube

1. In the Analysis Manager tree pane, expand the Analysis server and database in which you want to create the linked cube.

   Note The Analysis server cannot be the same as the server in the linked cube's data source.

2. Right-click the Cubes folder, and then click **New Linked Cube**.

3. In the **Linked Cube** dialog box, in the **Name** box, type a name for the linked cube.

4. In the **Source Cube** box, expand the data source that contains the source cube, and then select the source cube.

5. If you want to process the linked cube now, select the **Process after creating new cube** check box. The linked cube must be processed before end users can connect to it.

See Also

Linked Cubes

Analysis Services

How to process a linked cube

To process a linked cube

1. In the Analysis Manager tree pane, under the database that contains the linked cube, expand the Cubes folder.

2. Right-click the linked cube, and then click **Process**.

3. In the **Process a Cube** dialog box, click **Process,** and then click **OK**.

4. In the **Process** dialog box, wait for the linked cube to finish processing, or click **Stop** to halt and cancel processing.

Note Newly processed linked cubes are visible to end users only after they reconnect to the server computer.

See Also

Linked Cubes

Managing Linked Cubes

Analysis Services

Enhancing Dimensions

The following topics describe procedures for enhancing dimensions with optional features:

- How to create a member property in a shared dimension
- How to create a custom rollup operator for a shared dimension
- How to create a custom rollup operator for a private dimension
- How to create or select a column to store custom member formulas
- How to create a custom member formula in a write-enabled dimension
- How to create member groups
- How to create a dimension with multiple hierarchies in Dimension Editor
- How to create a dimension with multiple hierarchies in the Dimension Wizard
- How to add a hierarchy to an existing dimension

See Also

Enhancing Dimensions with Optional Features

Analysis Services

How to create a member property in a shared dimension

To create a member property in a shared dimension

1. In the Analysis Manager tree pane, expand the database that contains the dimension, and then expand the Shared Dimensions folder.

2. Right-click the shared dimension in which you want to create a member property, and then click **Edit**.

3. In Dimension Editor, right-click the level in which you want to create a member property, and then click **New Member Property**.

4. In the **Insert Member Property** dialog box, click the column that stores the member property values, and then click **OK**.

See Also

Creating Member Properties

Analysis Services

How to create a custom rollup operator for a shared dimension

To create a custom rollup operator for a shared dimension

1. Access the dimension that will contain the custom rollup operator by right-clicking the dimension, and then click **Edit**.

2. In Dimension Editor, in the tree pane, click the level for which to create the custom rollup operator.

3. If the properties pane is not expanded, expand it by clicking **Properties** beneath the tree pane.

4. In the properties pane, click the **Advanced** tab.

5. Click the value beside **Unary Operators**, and then click the edit (**...**) button.

6. In the **Define Unary Operator Column** dialog box, select the **Enable Unary Operators** check box to enable custom rollup operators for the level.

7. Create or select an existing column to store the custom rollup operators:

    - To create a new column in the dimension table, select **Create a new column**, and then in the **New column name** box, type the name of the new column.

      A dimension table can have multiple columns (one per level) that store custom rollup operators. Therefore, it is recommended that the new column name identify the column that stores the members to which the custom rollup operators apply (that is, the column for the level selected in Step 2.) For example, if in Step 2 you selected the Store Country level, and its members are stored in the **store_country** column, in the **New column name** box, type:

store_country_custom_rollup_operator

Note If the dimension is not write-enabled, you must use a tool other than Dimension Editor or Analysis Manager to add values to the new column.

- To select an existing column in the dimension table, select **Use an existing column**, and then in the **Existing column** box, select the column.

8. In the **Define Unary Operator Column** dialog box, click **OK**.

9. On the **File** menu, click **Save**.

   After you perform this procedure once for a level, you do not need to repeat it as long as the column that stores the custom rollup operators remains in the dimension table.

10. (Optional.) To browse your custom rollup operators, click the **Data** tab, and then expand the dimension members pane. Custom rollup operators are indicated next to member names. If you created a new, unpopulated column in Step 7, the plus operator (+), which is the default rollup operator, will display beside the member names.

11. (Optional.) To edit custom rollup operators in a write-enabled parent-child dimension, select a dimension member from the dimension members pane, and then select an operator in the column beside UNARY_OPERATOR in the member properties pane.

    - After editing, click **Save** on the **File** menu to commit changes to the dimension table.

      Note Before you can edit the write-enabled dimension, it must be included in a cube, and then the cube must be processed.

Analysis Services

How to create a custom rollup operator for a private dimension

To create a custom rollup operator for a private dimension

1. Access the cube with the private dimension that will contain the custom rollup operator by right-clicking the cube, and then click **Edit**.

2. In Cube Editor, in the tree pane, click the level for which to create the custom rollup operator.

3. If the properties pane is not expanded, expand it by clicking **Properties** beneath the tree pane.

4. In the properties pane, click the **Advanced** tab.

5. Click the value beside **Unary Operators**, and then click the edit (**...**) button.

6. In the **Define Unary Operator Column** dialog box, select the **Enable Unary Operators** check box to enable custom rollup operators for the level.

7. Create or select an existing column to store the custom rollup operators:

    - To create a new column in the dimension table, select **Create a new column**, and then in the **New column name** box, type the name of the new column.

      A dimension table can have multiple columns (one per level) that store custom rollup operators. Therefore, it is recommended that the new column name identify the column that stores the members to which the custom rollup operators apply (that is, the column for the level selected in Step 2.) For example, if in Step 2 you selected the Store Country level, and its members are stored in the **store_country** column, in the **New column name** box, type:

store_country_custom_rollup_operator

Note If the dimension is not write-enabled, you must use a tool other than Dimension Editor or Analysis Manager to add values to the new column.

- To select an existing column in the dimension table, select **Use an existing column**, and then in the **Existing column** box, select the column.

8. In the **Define Unary Operator Column** dialog box, click **OK**.

9. On the **File** menu, click **Save**.

   After you perform this procedure once for a level, you do not need to repeat it as long as the column that stores the custom rollup operators remains in the dimension table.

10. (Optional.) To browse custom rollup operators stored in an existing column, click the **Schema** tab, right-click the dimension table containing the stored operator, and then click **Browse Data**. Custom rollup operators are indicated in the respective column indicated in Step 7 for the first 1000 rows of data.

11. (Optional.) To edit custom rollup operators of a writable dimension, right-click the dimension in the Cube Editor tree pane and click **Browse** to display Dimension Browser. In the Dimension Browser tree pane, select the dimension member that has the custom rollup operator that you want changed. Click the value next to UNARY_OPERATORS in the member properties pane to modify the value for the selected member.

Analysis Services

How to create or select a column to store custom member formulas

To create or select a column to store custom member formulas

1. Access the dimension that will contain the custom member formulas:
    - If the dimension is shared, in the Analysis Manager tree pane, right-click the dimension, and then click **Edit**.
    - If the dimension is private, in the Analysis Manager tree pane, right-click the cube that contains the dimension, and then click **Edit**. In the Cube Editor tree pane, expand the dimension to reveal its levels.
2. In the editor tree pane, select the level for which you want to create custom member formulas.
3. If the properties pane is not expanded, expand it by clicking **Properties** beneath the tree pane.
4. In the properties pane, click the **Advanced** tab.
5. Click the value beside **Custom Members**, and then click the edit (**...**) button.
6. In the **Define Custom Member Column** dialog box, select **Enable Custom Members**.
7. Create or select a column to store the custom member formulas:
    - To create a new column in the dimension table, select **Create a new column**, and then in the **New column name** box, type the name of the new column.

A dimension table can have multiple columns (one per level) that store custom member formulas. Therefore, it is recommended that the new column name identify the column that stores the members to which the custom member formulas apply (that is, the column for the level selected in Step 2.) For example, if in Step 2 you selected the Store Country level, and its members are stored in the **store_country** column, in the **New column name** box, type:

```
store_country_custom_member_formula
```

- To select an existing column in the dimension table, select **Use an existing column**, and then in the **Existing column** box, select the column.

  Caution If you select an existing column, its contents will be overwritten by the custom member formulas.

8. In the **Define Custom Member Column** dialog box, click **OK**.

9. On the **File** menu, click **Save**.

After you perform this procedure once for a level, you do not need to repeat it as long as the column that stores the custom member formulas remains in the dimension table.

See Also

Creating Custom Member Formulas

Analysis Services

How to create a custom member formula in a write-enabled dimension

To create a custom member formula in a write-enabled dimension

1. (Only parent-child dimensions can be write-enabled.) Access the dimension that will contain the custom member formula:
    - If the dimension is shared, in the Analysis Manager tree pane, right-click the dimension, and then click **Edit**. In Dimension Editor, on the **View** menu, click **Data**.
    - If the dimension is private, in the Analysis Manager tree pane, right-click the cube that contains the dimension, and then click **Edit**. In the Cube Editor tree pane, right-click the dimension, and then click **Browse**.
2. In the dimension members pane, select the member for which you want to create the custom member formula.
3. In the custom member formula pane, create the custom member formula. Use any combination of the following methods:
    - Type.
    - Click the arithmetic operator buttons and parentheses buttons.
    - Click the edit (**...**) button to access MDX Builder in which you can construct the custom member formula with drag and drop techniques. After you are done in MDX Builder, click **OK**.
4. If the dimension is private, in Dimension Browser, click **Close**.
5. In the editor, on the **File** menu, click **Save**.

See Also

Creating Custom Member Formulas

Analysis Services

How to create member groups

To create member groups

1. Access the dimension that will contain the member groups:
    - If the dimension is shared, in the Analysis Manager tree pane, right-click the dimension, and then click **Edit**.
    - If the dimension is private, in the Analysis Manager tree pane, right-click the cube that contains the dimension, and then click **Edit**. In the Cube Editor tree pane, expand the dimension to reveal its levels.
2. Create a copy of the level that contains the members that will be the children of the member groups. Create it immediately above the original. To do this, follow these steps:
    a. In the editor tree pane, select the level to be copied.

    b. If the properties pane is not expanded, expand it by clicking **Properties** beneath the tree pane.

    c. In the properties pane, click the **Basic** tab.

    d. Record the value of the **Member Key Column** property.

    e. On the **Insert** menu, click **Level**.

    f. In the **Select Column** dialog box, select the column recorded in Step d, and then click **OK**.

    g. Select the new level.

    h. Drag the new level to a position immediately above the original level.

    i. In the properties pane, ensure that all of the new level's properties except **Name** have the same values as the original level.

3. In the editor tree pane, select the level in which you want to create member groups. (Select the new level from Step 2, not the original level.)

4. In the properties pane, click the **Advanced** tab.

5. For the **Grouping** property, select **Automatic**.

6. If you want to hide from end users the level that contains the member groups, follow these steps:
    a. In the editor tree pane, select the level that contains the member groups.

    b. In the properties pane, click the **Advanced** tab.

    c. Change the value of the **Visible** property to **False**.
7. In the editor, on the **File** menu, click **Save**.

See Also

Creating Member Groups

Analysis Services

How to create a dimension with multiple hierarchies in the Dimension Wizard

To create a dimension with a single defined hierarchy using the Dimension Wizard

1. In the Analysis Manager tree pane, expand the database in which you want to create a dimension with multiple hierarchies.

2. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Wizard**.

3. In the second step of the wizard select either **Star Schema: A single dimension table** or **Snowflake Schema: Multiple, related dimension tables**.

4. Follow the remaining wizard steps to define levels and various options for the dimension.

5. In the **Finish** step of the wizard, enter a name in the **Dimension name** box.

6. Select the **Create a hierarchy of a dimension** box.

7. Enter a name in the **Hierarchy name** box.

8. Click **Finish** to complete the wizard. After you complete the wizard, Dimension Editor appears so that you can further refine the dimension.

9. (Optional.) To create another hierarchy of the dimension, from the **File** menu in Dimension Editor, point to **New Dimension**, and then click

Wizard. Follow the steps in the next procedure, "To create a dimension with additional defined hierarchies using the Dimension Wizard," beginning with Step 3.

To create a dimension with additional defined hierarchies using the Dimension Wizard

1. In the Analysis Manager tree pane, expand the database in which you want to define additional hierarchies for a dimension with at least one named hierarchy.

2. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Wizard**.

3. In the second step of the Dimension Wizard select either **Star Schema: A single dimension table** or **Snowflake Schema: Multiple, related dimension tables**.

4. Follow the remaining wizard steps to define levels and various options for the dimension.

5. Select the **Create a hierarchy of a dimension** box.

6. Select a dimension name having a defined hierarchy from the **Dimension name** box.

7. Enter a name in the **Hierarchy name** box.

8. Click **Finish** to complete the wizard. After you complete the wizard, Dimension Editor appears so that you can further refine the dimension.

9. (Optional.) To create another hierarchy of the dimension, from the **File** menu in Dimension Editor, point to **New Dimension**, and then click

Wizard. Repeat Steps 3 through 8.

See Also

Creating Dimensions with Multiple Hierarchies

Dimension Hierarchies

Analysis Services

How to create a dimension with multiple hierarchies in Dimension Editor

To create a dimension with multiple hierarchies using Dimension Editor

1. In the Analysis Manager tree pane, expand the database in which you want to create the dimension with multiple hierarchies.

2. Right-click the Shared Dimensions folder, point to **New Dimension**, and then click **Editor**.

   In the **Choose a Dimension Table** dialog box, expand a data source, click the dimension table for the dimension, and then click **OK**. Dimension Editor appears with the dimension table showing in the **Schema** tab.

3. To add more tables to the dimension, on the **Insert** menu, click **Tables**, and then use the **Select Table** dialog box.

4. If the dimension contains multiple tables, ensure each is joined to another. To join two columns, in the **Schema** tab, drag one column to the other.

5. Create levels in the dimension. It is easiest to create the highest, most general level first and then create progressively lower, more specific levels; otherwise, if member counting is enabled, a confirmation dialog box appears each time you add a level. For each level you want to create:

   a. On the **Insert** menu, click **Level**.

   b. In the **Insert Level** dialog box, select the column on which the level is based, and then click **OK**. As you create each level, it appears in the tree pane.

6. (Optional.) In the properties pane, modify the properties of the dimension and its levels. For information about these properties, see Properties Pane (Dimension Editor Schema View).

7. To save the dimension, on the **File** menu, click **Save**, type a name for the dimension in the **New Dimension Name** dialog box. To define a hierarchy for the dimension, this name must be of the form *dimensionname.hierarchyname*, where *dimensionname* is unique if there are no existing hierarchies or the same as another if it is another hierarchy of a dimension; *hierarchyname* is unique for each hierarchy being defined. Click **OK**.

8. (Optional.) To create another hierarchy, repeat Steps 2 through 8.

See Also

Creating Dimensions with Multiple Hierarchies

Dimension Hierarchies

Analysis Services

How to add a hierarchy to an existing dimension

To add a hierarchy to a dimension with a previously identified hierarchy

1. In the Analysis Manager tree pane, expand the database in which you want to create the dimension with multiple hierarchies, and then expand the Shared Dimensions folder.

2. Right-click the shared dimension with an identified hierarchy in which you want to add a hierarchy, and then click **Edit**. Such dimensions have names where a period is used to separate the dimension name part from the hierarchy name part.

3. In Dimension Editor, edit the dimension by adding or deleting levels or changing its schema. For more information, see Creating a Shared Dimension with the Editor.

4. After editing the dimension, to save the dimension with a new hierarchy, on the **File** menu, click **Save As** to display the **Dimension: New Name** dialog box. To add a new hierarchy, change the hierarchy part of the name following the period in the **New name** box. (Do not change the dimension name part before the period.) Click **OK**.

   Important When additional hierarchies are created for dimensions, the hierarchy part of the name should not equal any current or future level name or member name in the dimension because queries using the dimension may be ambiguous.

5. (Optional.) To create another hierarchy, repeat Steps 2 through 5.

To add a hierarchy to a dimension without a previously identified hierarchy

1. In the Analysis Manager tree pane, expand the database in which you want to create the dimension with multiple hierarchies, and then expand the Shared Dimensions folder.

2. Right-click the shared dimension with an unidentified hierarchy in which you want to add a hierarchy, and then click **Edit**. The names of dimensions with unidentified hierarchies are those that do not contain periods.

3. In Dimension Editor, to create a new dimension with a new hierarchy based on the selected dimension, on the **File** menu, click **Save As** to display the **Dimension: New Name** dialog box. In the **New name** box, provide a different dimension name followed by a period and a hierarchy name. Click **OK**.

   IMPORTANT When creating dimensions with multiple hierarchies, the hierarchy part of the name should not equal any current or future level name or member name in the dimension because queries using the dimension may be ambiguous.

4. (Optional.) In Dimension Editor, edit the hierarchy of the dimension by adding or deleting levels or changing its schema. For more information, see Creating a Shared Dimension with the Editor.

5. After editing the dimension and its hierarchy, on the **File** menu, click **Save** to save changes.

6. (Optional.) To create additional hierarchies for the dimension, on the **File** menu, click **Save As** to display the **Dimension: New Name** dialog box. To add a new hierarchy, change the hierarchy part of the name following the period in the **New name** box. (In this case, do not change the dimension name part before the period.) Click **OK**.

7. Repeat Steps 4 through 6 to edit and save the new hierarchy.

See Also

Adding Hierarchies to Existing Dimensions

Dimension Hierarchies

Analysis Services

Enhancing Cubes with Optional Features

The following topics describe procedures for enhancing cubes with optional features. Topics are grouped beneath the feature they support.

- Calculated cells
    - How to create calculated cells
    - How to import calculated cells into a virtual cube
    - How to edit a calculated cells definition
- Calculated members
    - How to rename a calculated member
    - How to create a calculated member in a regular cube
    - How to import a calculated member into a virtual cube
    - How to edit a calculated member in a virtual cube
    - How to create a calculated member in a virtual cube
- Named sets
    - How to create a named set in a regular cube
    - How to import a named set into a virtual cube
    - How to edit a named set in a virtual cube
    - How to create a named set in a virtual cube

- Actions
    - How to create an action in a regular cube
    - How to import an action into a virtual cube
    - How to edit an action in a virtual cube
    - How to create an action in a virtual cube
    - How to test an action
- Write-enabled cubes
    - How to write-enable a cube
    - How to browse writeback data for a cube
    - How to delete writeback data for a cube and write-disable it
    - How to convert a cube's writeback data to a partition and write-disable the cube
- Drillthrough
    - How to specify drillthrough options for a regular cube
    - How to specify drillthrough options for a virtual cube
    - How to specify drillthrough options for a linked cube
    - How to specify drillthrough options for a partition

See Also

Enhancing Cubes with Optional Features

Analysis Services

How to create calculated cells

To create calculated cells

1. In the Analysis Manager tree pane, under the database to which you want to add calculated cells, expand the Cubes folder.

2. Right-click the cube or virtual cube to which you want to add calculated cells, and then click **Edit**.

3. In Cube Editor or Virtual Cube Editor, on the **Insert** menu, click **Calculated Cells**.

4. Follow the steps in the wizard.

After you finish, the new calculated cells definition appears under Calculated Cells in the tree pane of Cube Editor or Virtual Cube Editor.

See Also

Calculated Cells

How to edit a calculated cells definition

How to import calculated cells into a virtual cube

Analysis Services

How to import calculated cells into a virtual cube

To import calculated cells into a virtual cube

1. In the Analysis Manager tree pane, under the database that contains the calculated cells, expand the Cubes folder.

2. Right-click the virtual cube into which you want to import calculated cells, and then click **Edit**.

3. In Virtual Cube Editor, right-click the Calculated Cells folder, and then click **Import Calculated Cells**.

4. Under **Available Calculated Cells**, select the check boxes beside the calculated cells definitions that you want to import. If you want to import all the calculated cells definitions in a cube, select the check box next to the cube.

   To import a calculated cells definition that is not in the component cubes of the virtual cube, clear the **Show source cubes only** check box.

After you finish, the imported calculated cells definitions appear under the Calculated Cells folder in the tree pane of Virtual Cube Editor.

See Also

Calculated Cells

How to create calculated cells

How to edit a calculated cells definition

Analysis Services

How to edit a calculated cells definition

To edit a calculated cells definition

1. In the Analysis Manager tree pane, under the database that contains the calculated cells, expand the Cubes folder.

2. Right-click the cube or virtual cube containing the calculated cells definition, and then click **Edit**.

3. In the tree pane of Cube Editor or Virtual Cube Editor, expand the Calculated Cells folder.

4. Right-click the calculated cells definition you want to edit, and then click **Edit**.

5. Follow the steps in the Calculated Cells Wizard to make the changes to the calculated cells definition.

See Also

Calculated Cells

How to create calculated cells

How to import calculated cells into a virtual cube

Analysis Services

How to rename a calculated member

To rename a calculated member

1. In the Analysis Manager tree pane, under the database that contains the calculated member, expand the Cubes folder.

2. Right-click the cube that contains the calculated member, and then click **Edit**.

3. In the Cube Editor tree pane, right-click the calculated member, and then click **Rename**.

4. In the box next to the calculated member icon, type a new name, and then click outside the box.

See Also

Creating Calculated Members

Analysis Services

How to create a calculated member in a regular cube

To create a calculated member in a regular cube

1. In the Analysis Manager tree pane, under the database in which you want to create the calculated member, expand the Cubes folder.

2. Right-click the cube in which you want to create the calculated member, and then click **Edit**.

3. In Cube Editor, on the **Insert** menu, click **Calculated Member**.

4. In Calculated Member Builder, in the **Parent dimension** box, select the dimension that will include the calculated member, or select **Measures**.

5. In the **Parent member** box, specify the member that will include the calculated member. Click **Change** to select a member other than the displayed member. (**Change** is unavailable if you selected a one-level dimension or **Measures** in Step 4.)

6. In the **Member name** box, type a name for the calculated member.

7. In the **Value expression** box, construct an expression to produce the values of the calculated member. Use any combination of the following methods to add to the expression:

    - Drag items from the **Data** and **Functions** boxes.

    - Click an item in the **Data** or **Functions** box, and then click **Insert**.

- Click the arithmetic operator and number buttons.
- Type. This method is required to add functions from libraries other than the Microsoft® SQL Server™ 2000 Analysis Services function library.

8. (Optional.) To register additional function libraries, click **Register**.
9. To close Calculated Member Builder, click **OK**.
10. To save the cube with the new calculated member, in Cube Editor, on the **File** menu, click **Save**.

Example

To manually enter a calculated member that finds the average profit margin by store, which is defined as **1 - (Store Cost/Store Sales)**, follow these steps:

1. In the **Value expression** box, type **1-**.
2. Click the opening parenthesis (**(**) operator button, and then in the **Value expression** box, click immediately to the right of the parenthesis to place the cursor.
3. In the **Data** box, expand **Measures** and **MeasuresLevel** to display the individual measures.
4. Select **Store Cost**, and then click **Insert**.
5. In the **Value expression** box, place the cursor at the end of the phrase that was just entered, and then type a slash mark (/).
6. In the **Data** box, select **Store Sales**.

7. Click **Insert**.

8. Click the closing parenthesis (**)**) operator button.

   The following expression appears in the **Value expression** box:

   1-([Measures].[Store Cost]/[Measures].[Store Sales])

9. In the **Member Name** box, type **Average Store Margin**, and then click **OK**.

See Also

Creating Calculated Members in Regular Cubes

Analysis Services

How to import a calculated member into a virtual cube

To import a calculated member into a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In Virtual Cube Editor, on the **Edit** menu, point to **Import**, and then click **Calculated Members**.

3. In the **Import Calculated Members** dialog box, select the check box beside the calculated member, and then click **OK**. (To select all the calculated members in a cube, select the check box beside the cube.)

4. In Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Importing a Calculated Member into a Virtual Cube

Analysis Services

How to edit a calculated member in a virtual cube

To edit a calculated member in a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.
2. In the Virtual Cube Editor tree pane, right-click the calculated member, and then click **Edit**.
3. In Calculated Member Builder, in the **Parent dimension** box, select the dimension that will include the calculated member, or select **Measures**.
4. In the **Parent member** box, specify the member that will include the calculated member. Click **Change** to select a member other than the displayed member. (**Change** is unavailable if you selected a one-level dimension or **Measures** in Step 3.)
5. In the **Value expression** box, construct an expression to produce the values of the calculated member. Use any combination of the following methods to add to the expression:
    - Drag items from the **Data** and **Functions** boxes.
    - Click an item in the **Data** or **Functions** box, and then click **Insert**.
    - Click the arithmetic operator and number buttons.
    - Type.
6. (Optional.) To register additional function libraries, click **Register**.

7. To close Calculated Member Builder, click **OK**.

8. To save the calculated member, in Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Editing a Calculated Member in a Virtual Cube

Analysis Services

How to create a calculated member in a virtual cube

To create a calculated member in a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In Virtual Cube Editor, on the **Insert** menu, click **Calculated Member**.

3. In Calculated Member Builder, in the **Parent dimension** box, select the dimension that will include the calculated member, or select **Measures**.

4. In the **Parent member** box, specify the member that will include the calculated member. Click **Change** to select a member other than the displayed member. (**Change** is unavailable if you selected a one-level dimension or **Measures** in Step 3.)

5. In the **Member name** box, type a name for the calculated member.

6. In the **Value expression** box, construct an expression to produce the values of the calculated member. Use any combination of the following methods to add to the expression:

    - Drag items from the **Data** and **Functions** boxes.

    - Click an item in the **Data** or **Functions** box, and then click **Insert**.

    - Click the arithmetic operator and number buttons.

- Type.

7. (Optional.) To register additional function libraries, click **Register**.

8. To close Calculated Member Builder, click **OK**.

9. To save the calculated member, in Virtual Cube Editor, on the **File** menu, click **Save**.

Example

To manually enter a calculated member that finds the average profit margin by store, which is defined as **1 - (Store Cost/Store Sales)**, follow these steps:

1. In the **Value expression** box, type **1-**.

2. Click the opening parenthesis (**(**) operator button, and then in the **Value expression** box, click immediately to the right of the parenthesis to place the cursor.

3. In the **Data** box, expand **Measures** and **MeasuresLevel** to display the individual measures.

4. Select **Store Cost**, and then click **Insert**.

5. In the **Value expression** box, place the cursor at the end of the phrase that was just entered, and then type a slash mark (/).

6. In the **Data** box, select **Store Sales**.

7. Click **Insert**.

8. Click the closing parenthesis (**)**) operator button.

The following expression appears in the **Value expression** box:

```
1-([Measures].[Store Cost]/[Measures].[Store Sales])
```

9. In the **Member Name** box, type **Average Store Margin**, and then click **OK**.

See Also

Creating a Calculated Member in a Virtual Cube

Analysis Services

How to create a named set in a regular cube

To create a named set in a regular cube

1. In the Analysis Manager tree pane, under the database in which you want to create the named set, expand the Cubes folder.

2. Right-click the cube in which you want to create the named set, and then click **Edit**.

3. In Cube Editor, on the **Insert** menu, click **Named Set**.

4. In Named Set Builder, in the **Set name** box, type a name for the named set.

5. In the **Set expression** box, construct an expression for the named set. Use any combination of the following methods to add to the expression:

    - Drag items from the **Data** and **Functions** boxes.

    - Click an item in the **Data** or **Functions** box, and then click **Insert**.

    - Click the arithmetic operator and number buttons.

    - Type.

      Note If you create the set expression by explicitly naming the members in the set, enclose the list of members in a pair of braces ({}).

6. (Optional.) To register additional function libraries, click **Register**.

7. To close Named Set Builder, click **OK**.

8. To save the named set, in Cube Editor, on the **File** menu, click **Save**.

See Also

Creating Named Sets in Regular Cubes

Analysis Services

How to import a named set into a virtual cube

To import a named set into a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In Virtual Cube Editor, on the **Edit** menu, point to **Import**, and then click **Named Sets**.

3. In the **Import Named Sets** dialog box, select the check box beside the named set, and then click **OK**. (To select all the named sets in a cube, select the check box beside the cube.)

4. In Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Importing a Named Set into a Virtual Cube

Analysis Services

How to edit a named set in a virtual cube

To edit a named set in a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In the Virtual Cube Editor tree pane, right-click the named set, and then click **Edit**.

3. In Named Set Builder, in the **Set expression** box, construct an expression for the named set. Use any combination of the following methods to add to the expression:

    - Drag items from the **Data** and **Functions** boxes.

    - Click an item in the **Data** or **Functions** box, and then click **Insert**.

    - Click the arithmetic operator and number buttons.

    - Type.

      Note If you create the set expression by explicitly naming the members in the set, enclose the list of members in a pair of braces ({}).

4. (Optional.) To register additional function libraries, click **Register**.

5. To close Named Set Builder, click **OK**.

6. To save the named set, in Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Editing a Named Set in a Virtual Cube

Analysis Services

How to create a named set in a virtual cube

To create a named set in a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In Virtual Cube Editor, on the **Insert** menu, click **Named Set**.

3. In Named Set Builder, in the **Set name** box, type a name for the named set.

4. In the **Set expression** box, construct an expression for the named set. Use any combination of the following methods to add to the expression:

    - Drag items from the **Data** and **Functions** boxes.

    - Click an item in the **Data** or **Functions** box, and then click **Insert**.

    - Click the arithmetic operator and number buttons.

    - Type.

      Note If you create the set expression by explicitly naming the members in the set, enclose the list of members in a pair of braces ({}).

5. (Optional.) To register additional function libraries, click **Register**.

6. To close Named Set Builder, click **OK**.

7. To save the named set, in Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Creating a Named Set in a Virtual Cube

Analysis Services

How to create an action in a regular cube

To create an action in a regular cube

1. In the Analysis Manager tree pane, under the database in which you want to create the action, expand the Cubes folder.

2. Right-click the cube in which you want to create the action, and then click **Edit**.

3. In Cube Editor, on the **Insert** menu, click **Action**.

4. In the Action Wizard, in **Welcome** step, click **Next**.

5. In the **Select target** step, select the object to which the action is attached. Generally, in client applications, the action is displayed when end users select the target object; however, the client application determines which end-user operation displays actions. Click **Next**.

6. In the **Select the action type** step, select the type of action. The type indicates the kind of operation performed by the action. For more information about action types, see Creating Actions. Click **Next**.

7. In the **Define the Action Syntax** step, specify the parameters that are passed when the action is executed. The syntax must evaluate to a string. Click **Next**.

8. In the **Finish** step, specify the action name. Click **Finish**.

9. To save the action, in Cube Editor, on the **File** menu, click **Save**.

See Also

Creating Actions in Regular Cubes

How to test an action

Analysis Services

How to import an action into a virtual cube

To import an action into a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In Virtual Cube Editor, on the **Edit** menu, point to **Import**, and then click **Actions**.

3. In the **Import Actions** dialog box, select the check box beside the action, and then click **OK**. (To select all the actions in a cube, select the check box beside the cube.)

4. In Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Importing an Action into a Virtual Cube

How to test an action

Analysis Services

How to edit an action in a virtual cube

To edit an action in a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In the Virtual Cube Editor tree pane, right-click the action, and then click **Edit**.

3. In the Action Wizard, in the **Select target** step, select the object to which the action is attached. Generally, in client applications, the action is displayed when end users select the target object; however, the client application determines which end-user operation displays actions. Click **Next**.

4. In the **Select the action type** step, select the type of action. The type indicates the kind of operation performed by the action. For more information about action types, see Creating Actions. Click **Next**.

5. In the **Define the action syntax** step, specify the parameters that are passed when the action is executed. The syntax must evaluate to a string. Click **Next**.

6. In the **Finish** step, specify the action name. Click **Finish**.

7. To save the action, in Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Editing an Action in a Virtual Cube

How to test an action

Analysis Services

How to create an action in a virtual cube

To create an action in a virtual cube

1. In the Analysis Manager tree pane, right-click the virtual cube, and then click **Edit**.

2. In Virtual Cube Editor, on the **Insert** menu, click **Action**.

3. In the Action Wizard, in **Welcome** step, click **Next**.

4. In the **Select target** step, select the object to which the action is attached. Generally, in client applications, the action is displayed when end users select the target object; however, the client application determines which end-user operation displays actions. Click **Next**.

5. In the **Select the action type** step, select the type of action. The type indicates the kind of operation performed by the action. For more information about action types, see Creating Actions. Click **Next**.

6. In the **Define the action syntax** step, specify the parameters that are passed when the action is executed. The syntax must evaluate to a string. Click **Next**.

7. In the **Finish** step, specify the action name. Click **Finish**.

8. To save the action, in Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Creating an Action in a Virtual Cube

How to test an action

Analysis Services

How to test an action

To test an action in a processed regular or virtual cube

1. In the Analysis Manager tree pane, right-click the cube, and then click **Edit** or **Browse Data**. If you click **Edit**, select the **Data** tab in the editor.

2. If the target object such as a dimension is not visible in the data viewing pane as a column or row heading, drag it from the data slicing pane down to the data viewing pane.

3. In the data viewing pane, right-click the grid element that corresponds to the action's target object.

    - For the cube, right-click outside the grid area to select.

    - If the target is members of a dimension, right-click any member cell that is beneath the row or column heading corresponding to the highest level of the dimension.

    - If the target is a dimension object, right-click the column or row heading for the dimension's highest level to select the dimension.

    - If the target is members of a level, right-click any member cell that is beneath the row or column heading corresponding to the level name.

    - If the target is a level object, right-click the column or row heading for the level.

    - If the target is cells in the cube, right-click any data cell in the grid.

4. Point to the action to launch it.

   Note Actions based on named sets are not executable within Cube Browser, Cube Editor, or Virtual Cube Editor. They are, however, executable within custom applications built in a Component Object Model (COM) Automation language, such as Microsoft®Visual Basic®, that references Microsoft ActiveX® Data Objects (Multidimensional) (ADO MD) or in Microsoft Visual C++® using COM interfaces provided by the Microsoft SQL Server™ 2000 Analysis Services libraries.

See Also

Actions

Creating Actions

Cube Browser

Data Tab (Cube Editor Data View)

PivotTable Service

Analysis Services

How to write-enable a cube

To write-enable a cube

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, and then click **Write-enable**.

3. In the **Write-enable** dialog box, in the **Table name** box, type a name for the writeback table.

4. In the **Data source** box, select a data source name. To specify a new data source, follow these steps:
    a. Click **New**.

    b. In the **Data Link Properties** dialog box, specify the new data source, and then click **OK**. For more information, see Specifying Data Sources.
5. In the **Write-enable** dialog box, click **OK**.

See Also

Write-Enabling a Cube

Analysis Services

How to browse writeback data for a cube

To browse writeback data for a cube

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, point to **Writeback Options**, and then click **Browse Writeback Data**.

See Also

Browsing Writeback Data

Analysis Services

How to delete writeback data for a cube and write-disable it

To delete writeback data for a cube and write-disable it

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, point to **Writeback Options**, and then click **Disable Writeback**.

3. In the **Confirm Writeback Disable** dialog box, click **Yes**.

See Also

Deleting Writeback Data and Write-Disabling a Cube

Analysis Services

How to convert a cube's writeback data to a partition and write-disable the cube

To convert a cube's writeback data to a partition and write-disable the cube

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, point to **Writeback Options**, and then click **Convert to Partition**.

3. In the **Convert to Partition** dialog box, in the **Partition name** box, type a name for the partition.

4. Select an aggregation design option:

    - To design aggregations using the Storage Design Wizard, click **Design the aggregations for your partition now**. For more information, see Designing Storage Options and Aggregations.

    - To defer aggregation design, click **Design the aggregations later**.

    - To copy the aggregation design of an existing partition, click **Copy the aggregation design from an existing partition** and select the partition name from the **Copy from** list. If in the future you might merge the new partition with another, copy the aggregation design of the other partition. Merged partitions must have the same aggregation design.
5. To specify a filter (WHERE clause expression) that limits the data selected from the writeback table and added to the partition, click **Advanced**. For more information, see Partition Filters and Incremental

Update Filters.

6. To process the new partition, select the **Process the partition when finished** check box. Depending on the size of the writeback table, processing may take considerable time.

See Also

Converting Writeback Data to a Partition

Analysis Services

How to specify drillthrough options for a regular cube

To specify drillthrough options for a regular cube

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, and then click **Edit**.

3. On the **Tools** menu, click **Drillthrough Options**. (If this option is not available, the cube structure is not valid. You can try to determine the problem by clicking **Validate Cube Structure** on the **Tools** menu.)

4. In the **Drillthrough Options** dialog box, perform one of the following actions:

    - To enable drillthrough, select the **Enable drillthrough** check box.

    - To disable drillthrough, clear the **Enable drillthrough** check box. Go to Step 7.

5. In the **Columns** tab, select the columns to be displayed when drillthrough is executed.

    IMPORTANT If you select a column to which a cube role is denied access and then grant the cube role the ability to drill through, you create a security exposure. (You can deny access to a column by using dimension security or cell security. For more information, see Dimension Security and Cell Security.) When the end users in the cube role drill through, they access the denied column.

6. (Optional.) In the **Filter** tab, specify a filter (WHERE clause expression) to limit the result set returned by drillthrough. Do not include the keyword WHERE. For example, to limit the result set to

fourth-quarter data, specify:
"shipment_facts"."quarter"=4

For more information, see Drillthrough Filters.

7. Click **OK**.

8. In Cube Editor, on the **File** menu, click **Save**.

See Also

Specifying Drillthrough Options

Analysis Services

How to specify drillthrough options for a virtual cube

To specify drillthrough options for a virtual cube

1. In the Analysis Manager tree pane, under the database that contains the virtual cube, expand the Cubes folder.

2. Right-click one of the source cubes of the virtual cube, and then click **Edit**.

3. On the **Tools** menu, click **Drillthrough Options**. (If this command is not available, the cube structure is not valid. You can determine the problem by clicking **Validate Cube Structure** on the **Tools** menu.)

4. In the **Drillthrough Options** dialog box, ensure that the **Enable drillthrough** check box is selected. Drillthrough must be enabled on the source cube before you can enable drillthrough for the virtual cube.

5. In the **Columns** tab, select the columns to be displayed when drillthrough is executed.

   IMPORTANT If you select a column to which a cube role is denied access and then grant the cube role the ability to drill through, you create a security exposure. (You can deny access to a column by using dimension security or cell security. For more information, see Dimension Security and Cell Security.) When the end users in the cube role drill through, they access the denied column.

6. (Optional.) In the **Filter** tab, specify a filter (WHERE clause expression) to limit the result set returned by drillthrough. Do not include the keyword WHERE. For example, to limit the result set to fourth-quarter data, specify:
   ```
   "shipment_facts"."quarter"=4
   ```

For more information, see Drillthrough Filters.

7. (Optional.) If you do not want drillthrough to be enabled for the source cube after selecting the display columns, clear the **Enable drillthrough** check box. This option provides users the ability to drill through to the selected columns through the virtual cube but not through the current source cube.

8. Click **OK**.

9. In Cube Editor, on the **File** menu, click **Save**.

10. (Optional.) If there is more than one source cube, select the name of the next source cube from the **Cube** box within Cube Editor. Repeat Steps 3 through 9 as necessary for each source cube.

11. Exit Cube Editor.

12. In the Analysis Manager tree pane, right-click the virtual cube for which you want to enable drillthrough and click **Edit**. If the properties pane is not expanded in Virtual Cube Editor, expand it by clicking **Properties** beneath the tree pane.

13. Click the **Advanced** tab of the properties pane, and then click the value next to the **Enable Drillthrough** property.

14. Click **True** to enable drillthrough for the virtual cube.

15. In Virtual Cube Editor, on the **File** menu, click **Save**.

See Also

Cube Role Manager

Database Role Manager

Specifying Drillthrough Options

Analysis Services

How to specify drillthrough options for a linked cube

To specify drillthrough options for a linked cube based on a regular cube

1. On the publishing server hosting the source cube of the linked cube, in the Analysis Manager tree pane, under the database that contains the source cube, expand the Cubes folder.

2. Right-click the source cube, and then click **Edit**.

3. On the **Tools** menu, click **Drillthrough Options**. (If this command is not available, the cube structure is not valid. You can try to determine the problem by clicking **Validate Cube Structure** on the **Tools** menu.)

4. In the **Drillthrough Options** dialog box, perform one of the following actions:

    - To enable drillthrough, select the **Enable drillthrough** check box. Go to Step 5.

    - To disable drillthrough, clear the **Enable drillthrough** check box. Go to Step 7.

5. In the **Columns** tab, select the columns to be displayed when the end user attempts to drill through from the linked cube on the subscribing server.

    IMPORTANT If you select a column to which a cube role is denied access and then grant the cube role the ability to drill through, you create a security exposure. (You can deny access to a column by using dimension security or cell security. For more information, see Dimension Security and Cell Security.) When the end users in the cube role drill through, they access the denied column.

6. (Optional.) In the **Filter** tab, specify a filter (WHERE clause

expression) to limit the result set returned by drillthrough. Do not include the keyword WHERE. For example, to limit the result set to fourth-quarter data, specify:

```
"shipment_facts"."quarter"=4
```

For more information, see Drillthrough Filters.

7. Click **OK**.

8. In Cube Editor, on the **File** menu, click **Save**.

9. (Optional.) On the subscribing server, browse the linked cube from Analysis Manager to test the drillthrough settings for the linked cube.

To specify drillthrough options for a linked cube based on a virtual cube

1. On the publishing server hosting the virtual cube that is the source cube of the linked cube, in the Analysis Manager tree pane, under the database that contains the virtual cube, expand the Cubes folder.

2. Right-click a source cube of the virtual cube, and then click **Edit**.

3. On the **Tools** menu, click **Drillthrough Options**. (If this command is not available, the cube structure is not valid. You can determine the problem by clicking **Validate Cube Structure** on the **Tools** menu.)

4. In the **Drillthrough Options** dialog box, select the **Enable drillthrough** check box.

5. In the **Columns** tab, select the columns to be displayed when the end user drills through the linked cube on the subscribing server.

   IMPORTANT If you select a column to which a cube role is denied access and then grant the cube role the ability to drill through, you create a security exposure. (You can deny access to a column by using dimension security or cell security. For more information, see Dimension Security and Cell Security.) When the end users in the cube role drill through, they access the denied column.

6. (Optional.) In the **Filter** tab, specify a filter (WHERE clause expression) to limit the result set returned by drillthrough. Do not include the keyword WHERE. For example, to limit the result set to fourth-quarter data, specify:
   ```
   "shipment_facts"."quarter"=4
   ```

   For more information, see Drillthrough Filters.
7. (Optional.) If you do not want drillthrough to be enabled for the source cube on the publishing server after selecting the display columns, clear the **Enable drillthrough** check box. This option provides users the ability to drill through to the selected columns through the virtual cube on the publishing server and the linked cube on the subscribing server but not through the current source cube.
8. Click **OK**.
9. In Cube Editor, on the **File** menu, click **Save**.
10. (Optional.) If there is more than one source cube on the publishing server, select the name of the next source cube from the **Cube** box within Cube Editor. Repeat Steps 3 through 9 as necessary for each source cube.
11. Close Cube Editor.
12. On the publishing server in the Analysis Manager tree pane, right-click the virtual cube that is the source cube for the linked cube to enable drillthrough, and then click **Edit**.

13. If the properties pane is not expanded in Virtual Cube Editor, expand it by clicking **Properties** beneath the tree pane.

14. Click the **Advanced** tab of the properties pane, and then click the value next to the **Enable Drillthrough** property.

15. Click **True** to enable drillthrough for the virtual cube.

16. In Virtual Cube Editor, on the **File** menu, click **Save**.

17. (Optional.) On the subscribing server, browse the linked cube from Analysis Manager to test the drillthrough settings for the linked cube.

Analysis Services

How to specify drillthrough options for a partition

To specify drillthrough options for a partition

1. In the Analysis Manager tree pane, under the cube that contains the partition, expand the Partitions folder.

2. Right-click the partition, and then click **Edit**.

3. Advance to the **Finish** step of the Partition Wizard, and then click **Advanced**.

4. In the **Advanced Settings** dialog box, click **Drillthrough**.

5. In the **Columns** tab, select the columns to be displayed when drillthrough is executed.

   IMPORTANT If you select a column to which a cube role is denied access and then grant the cube role the ability to drill through, you create a security exposure. (You can deny access to a column by using dimension security or cell security. For more information, see Dimension Security and Cell Security.) When the end users in the cube role drill through, they access the denied column.

6. (Optional.) In the **Filter** tab, specify a filter (WHERE clause expression) to limit the result set returned by drillthrough. Do not include the keyword WHERE. For example, to limit the result set to fourth quarter data, specify:

   ```
   "shipment_facts"."quarter"=4
   ```

   For more information, see Drillthrough Filters.

7. Click **OK**.

8. In the **Advanced Settings** dialog box, click **OK**.

9. In the Partition Wizard, click **Finish**.

See Also

Specifying Drillthrough Options

Analysis Services

Updating Cubes and Dimensions

The following topics describe procedures for updating cubes and dimensions:

- How to incrementally update a cube
- How to refresh data in a cube
- How to incrementally update a shared dimension
- How to rebuild the structure of a shared dimension

See Also

Processing Cubes

Updating Cubes and Dimensions

Analysis Services

How to incrementally update a cube

To incrementally update a cube

Caution This procedure updates a partition. Incorrect use of partitions can result in inaccurate cube data. For more information, see Managing Partitions.

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, and then click **Process**.

3. In the **Process a Cube** dialog box, click **Incremental update**.

4. (Optional.) To incrementally update shared dimensions contained in the cube during processing, select **Incrementally update the shared dimensions used in this cube**.

5. Click **OK** to display the Incremental Update Wizard.

6. In the Incremental Update Wizard:
   a. In the **Welcome** step, click **Next**.

   b. If the cube contains multiple partitions, the **Select a partition to update** step appears. In the **Partition** box, select the partition to update, and then click **Next**.

   c. In the **Data source** box, select the data source that contains the data to add to the partition. You can select the same data source used by the partition or a different one. By default the same data source used by the partition is initially displayed. To select a different data source, click **Change**, select the data source, and then click **OK**. If you select a different data source, it must contain a fact table with the same structure and columns as the fact table for the partition, and it must contain dimension tables with the same structure and columns as the partition's dimension tables.

   d. In the **Fact table** box, select the table that contains the data to add to the partition. You can select the partition's fact table or a different table. By default the partition's fact table is initially displayed. If you select this table, you must use a filter, as described in the next step, to ensure that only data not already in the partition is added. To select a different table, click **Change**, select the table, and then click **OK**. If you select a different table, it must have the same structure and columns as the fact table for the partition. You must also manually merge the table with the fact table for the partition after the incremental update completes. For more information, see Fact Table Considerations When Merging Partitions. Click **Next**.

   e. Specify a filter (WHERE clause expression) to limit the data selected from the fact table and added to the partition. A filter is required if you select the fact table for the partition as the fact table for the incremental update (that is, if you select the default fact table). For more information, see Partition Filters and Incremental Update Filters. Click **Next**.

   f. Click **Finish**.

7. In the **Process** dialog box, wait for the incremental update to finish processing, or click **Stop** to halt and cancel processing.

See Also

Updating and Refreshing Cube Data

Analysis Services

How to refresh data in a cube

To refresh data in a cube

1. In the Analysis Manager tree pane, under the database that contains the cube, expand the Cubes folder.

2. Right-click the cube, and then click **Process**.

3. In the **Process a Cube** dialog box, click **Refresh data**.

4. (Optional.) To incrementally update shared dimensions contained in the cube during processing, select **Incrementally update the shared dimensions used in this cube**.

5. Click **OK**.

6. In the **Process** dialog box, wait for the data refresh to finish processing, or click **Stop** to halt and cancel processing.

See Also

Updating and Refreshing Cube Data

Analysis Services

How to incrementally update a shared dimension

To incrementally update a shared dimension

1. In the Analysis Manager tree pane, under the database that contains the shared dimension, expand the Shared Dimensions folder.

2. Right-click the shared dimension, and then click **Process**.

3. In the **Process a Dimension** dialog box, click **Incremental update**, and then click **OK**.

4. In the **Process** dialog box, wait for the incremental update to finish processing or click **Stop** to halt and cancel processing.

See Also

Updating and Rebuilding Shared Dimensions

Analysis Services

How to rebuild the structure of a shared dimension

To rebuild the structure of a shared dimension

1. In the Analysis Manager tree pane, under the database that contains the shared dimension, expand the Shared Dimensions folder.

2. Right-click the shared dimension, and then click **Process**.

3. In the **Process a Dimension** dialog box, click **Rebuild the dimension structure**, and then click **OK**.

4. In the **Process** dialog box, wait for the rebuild to finish processing or click **Stop** to halt and cancel processing.

After you process a shared dimension using the **Rebuild the dimension structure** option, you must process all of the cubes that include the dimension. For more information, see Processing Cubes.

See Also

Updating and Rebuilding Shared Dimensions

Analysis Services

Managing Data Mining Models

The following topic describes procedures for managing data mining models:

- How to start the Mining Model Wizard
- How to start Relational Mining Model Editor
- How to start OLAP Mining Model Editor
- How to process a data mining model
- How to start Mining Model Role Manager
- How to add a role to a data mining model

See Also

Mining Model Wizard

Relational Mining Model Editor

OLAP Mining Model Editor

Mining Model Role Manager

Analysis Services

How to start the Mining Model Wizard

To start the Mining Model Wizard

- In the Analysis Manager tree pane, under the database in which you want to create a data mining model, right-click the Mining Models folder, and then click **New Mining Model**.

After you complete the Mining Model Wizard, Relational Mining Model Editor or OLAP Mining Model Editor will appear, depending on whether you created a relational data mining model or an OLAP data mining model, respectively.

See Also

Mining Model Wizard

Relational Mining Model Editor

OLAP Mining Model Editor

Analysis Services

How to start Relational Mining Model Editor

To start Relational Mining Model Editor

1. In the Analysis Manager tree pane, under the database in which you have a relational data mining model, expand the Mining Models folder.

2. Right-click the relational data mining model to be edited, and then click **Edit**.

   Note If the structure of a relational data mining model is altered, the mining model must be processed before the mining model content can be viewed.

See Also

Relational Mining Model Editor

Analysis Services

How to start OLAP Mining Model Editor

To start OLAP Mining Model Editor

1. In the Analysis Manager tree pane, under the database in which you have an OLAP data mining model, expand the Mining Models folder.

2. Right-click the OLAP data mining model to be edited, and then click **Edit**.

   Note If the structure of an OLAP data mining model is altered, the mining model must be processed before the mining model content can be viewed.

See Also

OLAP Mining Model Editor

Analysis Services

How to process a data mining model

To process a data mining model from Analysis Manager

1. In the Analysis Manager tree pane, under the database in which you have a relational or OLAP data mining model, expand the Mining Models folder.

2. Right-click the relational or OLAP data mining model to be processed, and then click **Process**.

To process a data mining model from Relational Mining Model Editor

1. In the Analysis Manager tree pane, under the database in which you have a relational data mining model, expand the Mining Models folder.

2. Right-click the relational data mining model to be edited, and then click **Edit**.

3. In Relational Mining Model Editor, click the Tools menu, then click **Process Mining Model**.

To process a data mining model from OLAP Mining Model Editor

1. In the Analysis Manager tree pane, under the database in which you have an OLAP data mining model, expand the Mining Models folder.

2. Right-click the OLAP data mining model to be edited, and then click **Edit**.

3. In OLAP Mining Model Editor, click the Tools menu, then click **Process Mining Model**.

See Also

Relational Mining Model Editor

OLAP Mining Model Editor

Process a Mining Model Dialog Box

Analysis Services

How to start Mining Model Role Manager

To start Mining Model Role Manager

1. In the Analysis Manager tree pane, under the database in which you have a relational or OLAP data mining model, expand the Mining Models folder.

2. Right-click the relational or OLAP data mining model to be processed, and then click **Manage Roles**.

See Also

Mining Model Role Manager

Analysis Services

How to add a role to a data mining model

To add an existing role to a data mining model

1. In the Analysis Manager tree pane, under the database in which you have a relational or OLAP data mining model, expand the Mining Models folder.

2. Right-click the relational or OLAP data mining model to be processed, and then click **Manage Roles**.

3. Select the check box next to the name of the role to be added.

4. Click **OK**.

To add a new role to a data mining model

1. In the Analysis Manager tree pane, under the database in which you have a relational or OLAP data mining model, expand the Mining Models folder.

2. Right-click the relational or OLAP data mining model to be processed, and then click **Manage Roles**.

3. Click **New** to display the **Create a Mining Model Role** dialog box.

4. In the **Role Name** box, type the name of the new role.

5. (Optional.) In the **Description** box, type a description of the new role.

6. On the **Membership** tab, click **Add** to display the **Add Users and Groups** dialog box, where you can add users and groups to the new

role.

7. Click **OK** to return to **Mining Model Role Manager**.

8. Click **OK**.

IMPORTANT Any changes made to a data mining model role will propagate to the corresponding database role.

See Also

Mining Model Role Manager

Mining Model Role Dialog Box

Analysis Services

Archiving, Restoring, and Copying Data

The following topics describe procedures for archiving, restoring, and copying data:

- How to archive an Analysis Services database using Analysis Manager
- How to archive an Analysis Services database using the msmdarch command
- How to restore an Analysis Services database using Analysis Manager
- How to restore an Analysis Services database using the msmdarch command
- How to copy and paste an object

See Also

Archiving, Restoring, and Copying Data

Analysis Services

How to archive an Analysis Services database using Analysis Manager

To archive a Microsoft® SQL Server™ 2000 Analysis Services database using Analysis Manager

1. In the Analysis Manager tree pane, right-click the database, and then click **Archive Database**.

2. In the **Archive Database** dialog box:
   a. In the **Save in** box, specify a file name and path for the archive file.

   b. In the **Temp Folder** box, specify a folder in which temporary files can be created during the archive process.

   c. If you are archiving a database on a remote server, the **Remote Path** box is present. Specify the path of the Data directory containing the files for the database.

   d. Click **Archive**.
3. In the **Archive Database Progress** dialog box, monitor the progress of the archive process. You can cancel the archive process after it has begun. You can also save the archive log.

See Also

Archiving an Analysis Services Database

Analysis Services

How to archive an Analysis Services database using the msmdarch command

To archive a Microsoft® SQL Server™ 2000 Analysis Services database using the msmdarch command

1. Open a command prompt.

2. (Optional.) Use the **cd** command to access the directory containing the msmdarch.exe file. By default, this file is at the following location:

   C:\Program Files\Microsoft Analysis Services\Bin

3. Type a valid **msmdarch** command with the **/a** switch. The full syntax and an example are shown later in this topic.

4. Press ENTER.

Note You can use the **/?** switch to display a help window that shows the syntax of the switches.

Syntax

["*command-path*]**msmdarch**["] **/a** *Server* "*OLAPDataPath*" "*DatabaseName*" "*BackupFileName*" ["*LogFileName*" ["*TempDirectory*"]]

command-path

The path containing the msmdarch.exe file. By default, this path is:

C:\Program Files\Microsoft Analysis Services\Bin

Server

The server computer name that contains the database you want to archive.

OLAPDataPath

The path of the Data directory that contains the files for the database you want to archive. By default, this path is:

C:\Program Files\Microsoft Analysis Services\Data

DatabaseName

The name of the database you want to archive.

BackupFileName

The path, file name, and .cab extension of the archive file.

LogFileName

The path, file name, and .log extension of the archive log. If you specify an archive log that already exists, the new archive log is appended to it. If you specify an invalid path or file name, the archive log is written to the DBArchive.log file, which by default is C:\Program Files\Microsoft Analysis Services\Bin\DBArchive.log.

TempDirectory

The specified path of the temporary directory used for processing space. This option must be specified if *LogFileName* is specified.

Example

The following command archives the sample **FoodMart 2000** database included in Analysis Services.

```
"\Program Files\Microsoft Analysis Services\Bin\msmdarch" /a myser
"\Program Files\Microsoft Analysis Services\Data\" "FoodMart 2000"
"\My archives\server myserver\FoodMart 2000.cab"
```

See Also

Archiving an Analysis Services Database

Analysis Services

How to restore an Analysis Services database using Analysis Manager

To restore a Microsoft® SQL Server™ 2000 Analysis Services database using Analysis Manager

1. In the Analysis Manager tree pane, right-click the server to which you want to restore the database, and then click **Restore Database**.

2. If you are restoring a database on a remote server, the **Remote Server Data Directory** dialog box appears. Specify the path of the Data directory to which the database will be restored. This directory will contain the files for the database after the restoration is complete. Click **OK**.

3. In the **Open Archive File** dialog box, specify the path and file name of the archive file, and then click **Open**.

4. In the **Restore Database** dialog box, click **Restore**.

5. In the **Restore Database Progress** dialog box, monitor the progress of the restoration. You can cancel the restoration after it has begun. You can also save the restore log.

See Also

Restoring an Analysis Services Database

Analysis Services

How to restore an Analysis Services database using the msmdarch command

To restore a Microsoft® SQL Server™ 2000 Analysis Services database using the msmdarch command

1. Open a command prompt.

2. (Optional.) Use the **cd** command to access the directory containing the msmdarch.exe file. By default, this file is at the following location:

   C:\Program Files\Microsoft Analysis Services\Bin

3. Type a valid **msmdarch** command with the **/r** or **/rs** switch. The full syntax and an example are shown later in this topic.

4. Press ENTER.

Note You can use the **/?** switch to display a Help window that shows the syntax of the switches.

Syntax

["*command-path*]**msmdarch**["] *switch Server* "*OLAPDataPath*" "*BackupFileName*" ["*LogFileName*" ["*TempDirectory*"]]

command-path

(Optional.) The path containing the msmdarch.exe file. By default, this path is:

C:\Program Files\Microsoft Analysis Services\Bin

switch

Specifies the option to be used when executing the command. The **/r** switch is used to specify a regular restoration, which will overwrite the database if it exists. The **/rs** switch is used to restore the database from the archive file only if the database does not exist already.

Server

The server computer name to which you want to restore the database.

OLAPDataPath

The path of the Data directory that will contain the files for the database after the restoration is complete. By default, this path is:

C:\Program Files\Microsoft Analysis Services\Data

BackupFileName

The path, file name, and .cab extension of the archive file.

LogFileName

The path, file name, and .log extension of the restore log. If you specify a restore log that already exists, the new restore log is appended to it. If you specify an invalid path or file name, the restore log is written to the DBRestore.log file, which by default is C:\Program Files\Microsoft Analysis Services\Bin\DBRestore.log.

TempDirectory

The specified path of the temporary directory used for processing space. This option must be specified if *LogFileName* is specified.

Example

The following command restores the sample **FoodMart 2000** database included in Analysis Services.

```
"\Program Files\Microsoft Analysis Services\Bin\msmdarch" /r myserv
"\Program Files\Microsoft Analysis Services\Data\"
"\My archives\server myserver\FoodMart 2000.cab"
```

See Also

Restoring an Analysis Services Database

Analysis Services

How to copy and paste an object

To copy and paste an object

1. In the Analysis Manager tree pane, right-click the object, and then click **Copy**.

2. Right-click the target location for the new object, and then click **Paste**.

Copy is not available for all object types. **Paste** is not available if you select a target location that is invalid for the object type you have copied.

See Also

Copying and Pasting Objects

Analysis Services

Analyzing and Optimizing Performance

The following topics describe procedures for analyzing and optimizing performance:

- How to start the Usage Analysis Wizard
- How to start the Usage-Based Optimization Wizard

See Also

Analyzing and Optimizing Performance

Analysis Services

How to start the Usage Analysis Wizard

To start the Usage Analysis Wizard

1. In the Analysis Manager tree pane, under the database that contains the cube you want to analyze, expand the Cubes folder.

2. Right-click the cube you want to analyze, and then click **Usage Analysis**.

See Also

Analyzing Usage Patterns

Analysis Services

How to start the Usage-Based Optimization Wizard

To start the Usage-Based Optimization Wizard

1. In the Analysis Manager tree pane, under the database that contains the cube you want to optimize, expand the Cubes folder.

2. Right-click the cube you want to optimize, and then click **Usage-Based Optimization**.

See Also

Optimizing Performance Based on Usage

Analysis Services

Automating and Scheduling Administrative Tasks

The following topics describe procedures for automating and scheduling administrative tasks:

- How to create an Analysis Services Processing task
- How to create a Relational Data Mining Prediction Query task

See Also

Automating and Scheduling Administrative Tasks

Analysis Services

How to create an Analysis Services Processing task

To create an Analysis Services Processing task

1. Open DTS Designer:
    a. On your desktop click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.

    b. In the console tree, expand the server on which you want to create the Data Transformation Services (DTS) package that will contain the Analysis Services Processing task.

    c. Right-click the Data Transformation Services folder, and then click **New Package**.
2. In DTS Designer, from the **Task** toolbar, drag the icon for the Analysis Services Processing task onto the design sheet. This icon appears here.
3. (Optional.) In the **Analysis Services Processing Task** dialog box, in the **Description** box, type a task description.

4. (Optional.) To limit the tree to objects on the local Analysis server and set the task to process an object or objects on the server computer where the package is stored, select **Local server**. If you select this option, you can later migrate the package to another server computer and execute it to process an object or objects on that server computer. This processing requires that the other server computer stores the processed objects' meta data and has access to their data sources. Meta data can be copied easily from one server to another by using the **Copy** and **Paste** commands in the Analysis Manager tree pane.

5. In the **Analysis Services Processing Task** dialog box, expand the

server tree, and then select the object or folder to process.

6. Select a processing option. Selecting an option determines the value of the **ProcessOption** property, which is used in Microsoft® ActiveX® Script tasks to modify processing options in Analysis Services Processing tasks.

   For more information about processing options, see Processing Cubes and Dimension Processing. For more information about valid values of the **ProcessOption** property, see Changing Properties of an Analysis Services Processing Task.

7. If you selected the **Incremental update** option for a partition or single-partition cube, you can:

   - Change the data source and fact table used for the incremental update. Click the edit (**...**) button beside the **Fact table** box and use the **Choose a Fact Table** dialog box.

   - Specify a filter to limit the fact table records used in the incremental update. Click the edit (**...**) button beside the **Filter** box and use the **Filter Expression** dialog box.

   Note You must specify a filter if you select the default fact table. Otherwise, the cube or partition will contain duplicate, and therefore inaccurate, data.

   If you change the fact table, the new fact table must have the same structure and columns as the default fact table. You must also merge the new fact table with the default fact table after the incremental update is complete. For more information, see Fact Table Considerations When Merging Partitions.

8. Click **OK** to finish creating the task. To save the task in a DTS package in DTS Designer, on the **Package** menu, click **Save**. For information about the different ways you can save your package, see Saving a DTS Package.

See Also

ActiveX Script Task

Creating a Package with DTS Designer

Creating an Analysis Services Processing Task

Save DTS Package

Analysis Services

How to create a Relational Data Mining Prediction Query task

To create a Relational Data Mining Prediction Query task

1. Start DTS Designer:
    a. On your desktop click **Start**, point to **Programs**, point to **Microsoft SQL Server**, and then click **Enterprise Manager**.

    b. In the console tree, expand the server on which you want to create the package that will contain the Data Mining Prediction Query task.

    c. Right-click the Data Transformation Services folder, and then click **New Package**.
2. In DTS Designer, from the **Task** tool palette, drag the icon for the Data Mining Prediction Query task onto the DTS Designer design sheet. This icon appears here.
3. (Optional.) In the **Data Mining Prediction Query Task** dialog box, in the **Name** box, type a new name to replace the default name for the task.

4. (Optional.) In the **Description** box, type a task description. This description is used to identify the task in DTS Designer.

5. In the **Server** box, type the name of the Analysis server that contains the data mining model to be used as the source for the prediction query. The server name is the same as the computer name on the network. Do not use UNC or network paths.

6. From the **Database** list, select the database that contains the mining model to be queried.

7. If the mining model you want to use for the prediction query is not already highlighted in the **Mining Models** box, select a mining model from the box by clicking its name or icon. You can view some of the properties of the mining model in the **Details** box.

8. Click the **Query** tab, and then in the **Input data source** box, either type a valid ActiveX® Data Objects (ADO) connection string to the case table containing the input and predictable columns for the query, or click the edit (**...**) button to display the **Data Link Properties** dialog box, where you can build the connection string.

   In the **Prediction query** box, type the syntax, or click **New Query** to display Prediction Query Builder. The prediction query syntax must conform to the OLE DB for Data Mining specification. For more information about the OLE DB for Data Mining specification, see the Microsoft OLE DB Web page at the Microsoft Web site.

9. (Optional.) In the **Output table** box, type a new name for the output table to replace the default name.

10. Click **OK** to finish creating the task. To save the task in a DTS package in DTS Designer, on the **Package** menu, click **Save**. For information about the different ways you can save your package, see Saving a DTS Package.

See Also

ActiveX Script Task

Creating a Package with DTS Designer

Save DTS Package