SSAS Def

Friday, July 18, 2008 11:44 AM

Information about how the data is stored and structured is called ***metadata*.**

Fact

The centralized table in a star schema is called a fact table. A fact table typically has two types of columns: those that contain facts and those that are foreign keys to dimension tables. The primary key of a fact table is usually a composite key that is made up of all of its foreign keys. Fact tables contain the content of the data warehouse and store different types of measures like additive, non additive, and semi additive measures.

Measure Types

- Additive - Measures that can be added across all dimensions.
- Non Additive - Measures that cannot be added across all dimensions.
- Semi Additive - Measures that can be added across few dimensions and not with others.

A fact table might contain either detail level facts or facts that have been aggregated (fact tables that contain aggregated facts are often instead called summary tables).

Dimension

Dimension tables contain attributes that describe fact records in the fact table. Dimension tables contain hierarchies of attributes that aid in summarization. For example, a dimension containing Year would often contain a hierarchy that month , with each of these month further subdivided a until the lowest level is reached (i.e date)

Aggregation Summarized values of a measure

Cache Server-based storage locations both in memory (automatic) or on disk (designed) that enhance query performance

Calculated member A mechanism for aggregating measures using formulas more complex than those stored in a cube

Cube A collection of one or more related measure groups and their associated dimensions

Cube metadata Instructions for creating and querying OLAP structures such as cubes and dimensions

Hierarchy Levels of aggregation within a single dimension

Measure group The conceptual container of detail values from a single fact table, along with all possible aggregations for one or more dimension hierarchies

Online analytical processing (OLAP) : A database system optimized to support decision-making processes

Online transaction processing (OLTP) : A database system used to manage transactions such as order processing

Unified Dimensional Model (U The measure groups and dimensions that define your organization's BI data; essentially synonymous with a cube

Pasted from <file:///C:\Users\Zainab\Desktop\SSAS%20Note.doc>

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 **several 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.

The **Incrementally update the dimensions of this cube** option causes the cube's dimensions to be incrementally updated when the cube is processed

Pasted from <http://msdn.microsoft.com/en-us/library/aa933573(SQL.80).aspx>

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**.

Pasted from <http://msdn.microsoft.com/en-us/library/aa216366(SQL.80).aspx>

Dimensions

Degenerate Dimension

It is a dimension that is stored in the fact table rather than the dimension table. It eliminates the need to join to the Di mension table.

Examples:
1 An insurance claim line : fact table typically includes both **claim and policy numbers** as degenerate dimensions.

2 A manufacturer could include degenerate dimensions for the quote, order, and bill of lading numbers in the shipments fact table.

During design reviews, we sometimes find a dimension table growing proportionately with the fact table. As rows are inserted into the fact table, new rows are also inserted into a related dimension table, often at the same rate as rows were added to the fact table. This situ ation should send a red flag waving. Usually when a dimension table is growing at roughly the same rate as the fact table, there is a degenerate dimension lurking that has been missed in the initial design.

Parent Child Dimension

Dimension referring to itself

Conformed Dimension/Shared Dimension/Linked Dimension

Conformed dimension is a dimension that is shared across fact tables. (Product Dimension is shared between 2 Facts (Sales and Inventory))

Junk Dimension

It is a dimension that contains miscellaneous data (like flags and indicators) that do not fit in base Dimension table.

Role - Playing Dimension

The situation where a single physical dimension table appears several times in a single fact table. Each of the dimension rol es is represented as a separate logical table with unique column names through views.

Eg: Effective date and operational date for the policy

Audit dimension.
This dimension is designed to track data lineage and quality. Row counts from the source, destination and redirection error r ows.

Fact-less fact table

A fact table that has no facts but captures certain many-to-many relationships between the dimension keys. Most often used to represent events or provide coverage information that does not appear in other fact tables. Contains only surrogate keys from all the dimensions and no measure

Slowly Changing Dimension
Type 1 – Birth Date (erroneously entered birth date)

Type 2 – Zip Code (Key input to the insurance pricing and risk algorithm)
When the Zip Code changes, we create a new policyholder dimension row with a new surrogate key and updated geographic attribu te

Type 3 – First the management use to classify the policyholders as Residential, Commercial and Public entity but now the management de cides to classify them as Large Multinational, Medium and Small commercials and residential
So we add a new column to dimension “New Segment Type”

Multi-valued Dimension
E.g.: Commercial customers may be associated with one or more SIC (Standard Industry Classification) Codes
E.g.: Coverage Dimension can have a dozen of separate coverage product available to sell for a given type of covered items.

Data Warehouse

Sunday, August 03, 2008 7:01 PM

The Purpose of a Data Warehouse
A data warehouse is a repository for storing and analyzing numerical information. A data warehouse stores stable, verified data values. You might find it helpful to compare some of the most important differences between a data warehouse and a transaction database.

■ A transaction database helps people carry out activities, while a data warehouse helps people make plans. For example, a transaction database might show which seats are available on an airline flight so that a travel agent can book a new reservation. A data warehouse, on the other hand, might show the historical pattern of empty seats by flight so that an airline manager can decide whether to adjust flight schedules in the future.

■ A transaction database focuses on the details, while a data warehouse focuses on highlevel aggregates. For example, a parent purchasing the latest popular children's book doesn't care about inventory levels for the Juvenile Fiction product line, but a manager planning the rearranging of store shelving may be very interested in a general decline in sales of computer book titles (for subjects other than SQL Server 2005). The implication of this difference is that the core data in a warehouse are typically numeric values that can be summarized.

■ A transaction database is typically designed for a specific application, while a data warehouse integrates data from different sources. For example, your order processing application—and its database—probably includes detailed discount information for each order, but nothing about manufacturing cost overruns. Conversely, your manufacturing application—and its database—probably includes detailed cost information, but nothing about sales discounts. By combining the two data sources in a data warehouse, you can calculate the actual profitability of product sales, possibly revealing that the fully discounted price is less than the actual cost to manufacture. But no worries: You can make up for it in volume.
■ A transaction database is concerned with now; a data warehouse is concerned with activity over time. For example, in a simple bank account, each transaction—that is, each deposit or withdrawal—creates an instantaneous change in the account balance. The transaction system rarely maintains historical balances, and even transaction logs are usually archived after a month or two. In a data warehouse, you can store many years of
6 Part I: Getting Started with Analysis Services
transaction data (perhaps summarized), and you can also store snapshots of historical balances. This allows you to compare what you did today with what you did last month or last year. When making decisions, the ability to see a wide time horizon is critical for distinguishing between trends and random fluctuations.
■ A transaction database is volatile; its information constantly changes as new orders are placed or cancelled, as new products are built or shipped, or as new reservations are made. A data warehouse is stable; its information is updated at standard intervals—perhaps monthly, weekly, or even hourly—and, in an ideal world, an update would add values for the new time period only, without changing values previously stored in the warehouse.
■ A transaction database must provide rapid retrieval and updating of detailed information; a data warehouse must provide rapid retrieval of highly summarized information. Consequently, the optimal design for a transaction database is opposite to the optimal design for a data warehouse. In addition, querying a live transaction database for management reporting purposes would slow down the transaction application to an unacceptable degree.

MDX

Sunday, August 03, 2008 3:07 AM

Multidimensional Expressions (MDX)

Multidimensional Expressions (MDX) is the query language that you use to work with and retrieve multidimensional data in Microsoft SQL Server 2005 Analysis Services (SSAS). MDX is based on the XML for Analysis (XMLA) specification, with specific extensions for SQL Server 2005 Analysis Services.

Maximum

```
WITH MEMBER Measures.x AS Max
   (
       [Date].[Calendar].2003
      , [Measures].[Order Quantity]
   )
SELECT Measures.x ON 0
,NON EMPTY [Date].[Calendar].[Calendar Quarter]*
   [Product].[Product Categories].[Subcategory].members *
   [Geography].[Geography].[Country].Members
ON 1
FROM [Adventure Works]
```

SUM

```
WITH MEMBER Measures.NorthAmerica AS SUM
      (
         {[Geography].[Country].&[Canada]
            , [Geography].[Country].&[United States]}
       ,[Measures].[Reseller Sales Amount]
      )
SELECT {[Measures].[NorthAmerica]} ON 0,
[Product].[Category].members ON 1
FROM [Adventure Works]
```

DISTINCT

```
Select Distinct({[Geography].[Country].&[Australia],
[Geography].[Country].&[France],
[Geography].[Country].&[Canada],
[Geography].[Country].&[Australia],
[Geography].[Country].&[United States],
[Geography].[Country].&[United Kingdom],
[Geography].[Country].&[Canada]}) on 1,
[Measures].[Reseller Sales Amount] on 0
from [Adventure Works]
```

IF FUNCTION

```
IIf(Logical_Expression, Expression1, Expression2)
```

INTERSECT

```
Intersect({[1994], [1995], [1996]}, {[1995], [1996], [1997]})
```

returns the set {[1995], [1996]}.

Topic	Description
Key Concepts in MDX (MDX)	Describes key MDX concepts for working with SQL Server 2005 Analysis Services cubes.
MDX Syntax Elements (MDX)	Describes the various syntax elements available in the MDX language for MDX expressions, statements, and scripts.
MDX Query Fundamentals (MDX)	Describes MDX query fundamentals, including the MDX SELECT statement, axes, the FROM clause, the WHERE clause, named sets, calculated members, cell calculations, property values, and manipulating or modifying data.
MDX Scripting Fundamentals (MDX)	Describes how to construct and use MDX scripts within a SQL Server 2005 Analysis Services cube
MDX Language Reference (MDX)	Describes the statements, operators, and functions that define the MDX language.

Pasted from <http://msdn.microsoft.com/en-us/library/ms145506.aspx>

Overview

Sunday, August 03, 2008 1:34 AM

Microsoft SQL Server 2005 Analysis Services (SSAS) delivers online analytical processing (OLAP) and data mining functionality for business intelligence applications. Analysis Services supports OLAP by letting you design, create, and manage multidimensional structures that contain data aggregated from other data sources, such as relational databases. For data mining applications, Analysis Services lets you design, create, and visualize data mining models that are constructed from other data sources by using a wide variety of industry-standard data mining algorithms.

Analysis Services Architecture

Microsoft SQL Server 2005 Analysis Services (SSAS) uses both server and client components to supply online analytical processing (OLAP) and data mining functionality for business intelligence applications:

- The server component of Analysis Services is implemented as a Microsoft Windows service. SQL Server 2005 Analysis Services su pports multiple instances on the same computer, with each instance of Analysis Services implemented as a separate instance of the Windows service.
- Clients communicate with Analysis Services using the public standard XML for Analysis (XMLA), a SOAP -based protocol for issuing commands and receiving responses, exposed as a Web service. Client object models are also provided over XMLA, and can be accessed either by using a managed provider, such as ADOMD.NET, or a native OLE DB provider.
- Query commands can be issued using the following languages: SQL; Multidimensional Expressions (MDX), an industry standard que ry language for analysis; or Data Mining Extensions (DMX), an industry standard query language oriented toward data mining. Analysis Services Scripting Language (ASSL) can also be used to manage Analysis Services database objects.

Analysis Services also supports a local cube engine that enables applications on disconnected clients to browse locally stored multidimensional data.

OLAP

Working with Online Analytical Processing (OLAP)

Online analytical processing (OLAP) allows you to access aggregated and organized data from business data sources, such as data warehouses, in a multidimensional structure called a cube. Microsoft SQL Server 2005 Analysis Services (SSAS) provides tools and features for OLAP that you can use to design, deploy, and maintain cubes and other supporting objects. Before you start integrating cubes and other OLAP functionality into your business intelligence solutions, make sure you are familiar with the following concepts and decisions.

Working in Project Mode Versus Online Mode
You can define and instantiate OLAP objects directly in an Analysis Services database (called *online mode* or sometimes *direct connect mode*) or within an Analysis Services project (called *project mode*). Generally, you will develop your OLAP solution within an Analysis Services project.

Working with Relational Schemas
You generally define OLAP objects based on an existing data source, such as a data warehouse or production database. However, you can also define OLAP objects without a data source and then have the Schema Generation Wizard create the underlying relational schema based on the OLAP objects defined in an Analysis Services project.

Defining and Configuring Dimensions, Attributes, and Hierarchies
Dimensions, attributes, and hierarchies are OLAP objects that you define and configure at the Analysis Services database level. These OLAP objects exist independent of any OLAP cubes and can be used in one or more cubes. To a limited extent, these objects can be customized within each cube.

Defining and Configuring Cubes and Cube Properties
Cubes are OLAP objects consisting of related measures and dimensions that you configure within an Analysis Services database. You can define and configure multiple cubes within a single database and each cube can use some or all of the same dimensions. You can also define a single cube that contains multiple measure groups in the same database rather than defining separate cubes. When you define a cube with multiple measure groups, you need to define how dimensions relate to each measure group and customize, as appropriate, dimension objects within each cube and measure group. When defining a cube, you also define advance cube properties, including calculations, KPIs, actions, partitions and aggregations, perspectives and translations.

SSAS Objects

Analysis Services Objects
A Microsoft SQL Server 2005 Analysis Services (SSAS) instance contains database objects and assemblies for use with online analytical processing (OLAP) and data mining.

- Databases contain OLAP and data mining objects, such as data sources, data source views, cubes, measures, measure groups, dimensions, attributes, hierarchies, mining structures, mining models and roles.
- Assemblies contain user-defined functions that extend the functionality of the intrinsic functions provided with the Multidimensional Expressions (MDX) and Data Mining Extensions (DMX) languages.

In This Section
The following topics describe objects shared by both OLAP and data mining features in Analysis Services.

Topic	**Description**
Data Sources (Analysis Services)	Describes a data source in Analysis Services.
Data Source Views (Analysis Services)	Describes a logical data model based on one or more data sources, in Analysis Services.
Cubes (Analysis Services)	Describes cubes and cube objects, including measures, measure groups, dimension usage relationships, calculations, key performance indicators, actions, translations, partitions, and perspectives.
Dimensions (Analysis Services)	Describes dimensions and dimension objects, including attributes, attribute relationships, hierarchies, levels, and members.
Mining Structures (Analysis Services)	Describes mining structures and mining objects, including mining models.
Roles (Analysis Services)	Describes a role, the security mechanism used to control access to objects in Analysis Services.
Assemblies (Analysis Services)	Describes an assembly, a collection of user-defined functions used to extend the MDX and DMX languages, in Analysis Services.

Data Sources

Sunday, August 03, 2008 2:25 AM

Data Sources (Analysis Services)

A data source in Microsoft SQL Server 2005 Analysis Services (SSAS) represents a connection to a data source and contains the connection string that defines how Analysis Services connects to a physical data store using a managed Microsoft .NET Framework or native OLE DB provider. The connection string contains server name, database, security, timeout, and other connection-related information. Analysis Services directly supports many data sources. Supported data sources include Microsoft SQL Server databases and databases created by other products, including Oracle, DB2, and Teradata.

You can define a new data source or define a data source based on a previously defined data source.

Working with Data Sources (Analysis Services)

A Microsoft SQL Server 2005 Analysis Services (SSAS) data source is an object that provides the Analysis Services service with the information needed for it to connect to a source of information for the business intelligence solution. Analysis Services can access data from one or more sources of data, provided that Analysis Services is able to construct the OLAP or data mining queries required by the business intelligence solution. The list of providers that you can utilize in an Analysis Services project is increasing over time as Microsoft and third-party vendors provide support for SQL Server 2005 Analysis Services. Currently, the following providers and relational databases are supported in Analysis Services projects:

- SQL Server 7.0 using the SQL OLE DB Provider or the .NET native OLE DB provider (x86, x64, and ia64).
- SQL Server 2000 using the SQL OLE DB Provider or the .NET native OLE DB provider (x86, x64, and ia64).
- SQL Server 2005 using the SQL OLE DB Provider or the .NET native OLE DB provider (x86, x64, and ia64).
- Oracle 9.0 using the Microsoft OLE DB Provider for Oracle or the .NET native OLE DB provider (x86 only).
- IBM DB2 8.1 using Microsoft OLE DB Provider for DB2 (x86, x64, ia64) - only available for Microsoft SQL Server 2005 Enterprise Edition or Microsoft SQL Server 2005 Developer Edition and downloadable as part of the Feature Pack for Microsoft SQL Server 2005 Service Pack 1.
- Access with Microsoft Jet 4.0 OLE DB provider (x86 only).
- Teradata v2R6 with OLE DB 1.3 provider from NCR (x86 only).

Data Source Views

Sunday, August 03, 2008 2:28 AM

Cubes

Sunday, August 03, 2008 2:27 AM

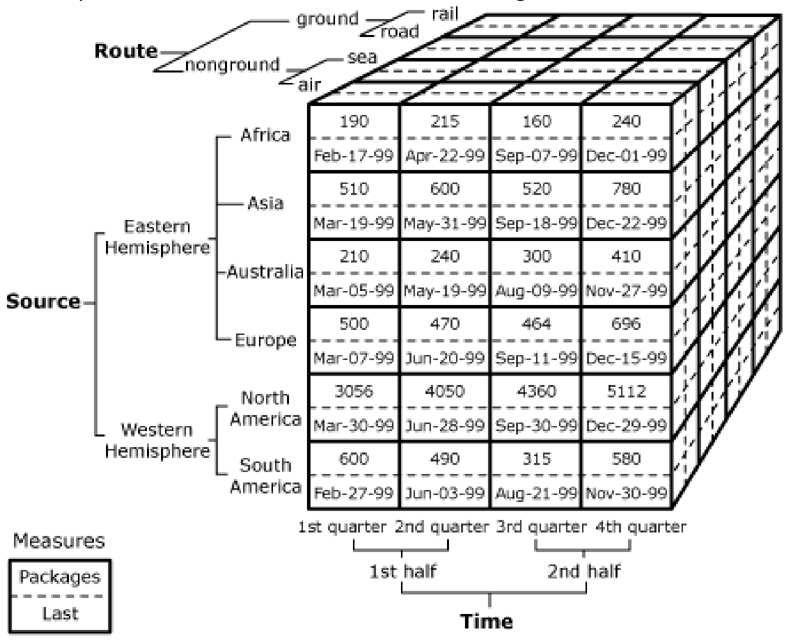
Cubes (Analysis Services)
A cube is a set of related measures and dimensions that is used to analyze data.

- A measure is a fact, which is a transactional value or measurement that a user may want to aggregate. Measures are sourced from columns in one or more source tables, and are grouped into measure groups.
- A dimension is a group of attributes that represent an area of interest related to the measures in the cube, and which are used to analyze the measures in the cube. For example, a Customer dimension might include the attributes Customer Name, Customer Gender, and Customer City, which would enable measures in the cube to be analyzed by Customer Name, Customer Gender, and Customer City. Attributes are sourced from columns in one or more source tables. The attributes within each dimension can be organized into hierarchies to provide paths for analysis.

A cube is then augmented with calculations, key performance indicators (KPIs), actions, partitions, perspectives, and translations. A cube is essentially synonymous with a Unified Dimensional Model (UDM).
A cube can be developed with or without an underlying relational data source.Facts in a cube are aggregated across dimensions, based on the dimension hierarchies.

Introduction to Cubes
A cube is defined by its measures and dimensions. The measures and dimensions in a cube are derived from the tables and views in the data source view on which the cube is based, or which is generated from the measure and dimension definitions.

Cube Example
The Imports cube contains two measures, Packages and Last, and three related dimensions, Route, Source, and Time.



	1st quarter	2nd quarter	3rd quarter	4th quarter
Africa	190 / Feb-17-99	215 / Apr-22-99	160 / Sep-07-99	240 / Dec-01-99
Asia	510 / Mar-19-99	600 / May-31-99	520 / Sep-18-99	780 / Dec-22-99
Australia	210 / Mar-05-99	240 / May-19-99	300 / Aug-09-99	410 / Nov-27-99
Europe	500 / Mar-07-99	470 / Jun-20-99	464 / Sep-11-99	696 / Dec-15-99
North America	3056 / Mar-30-99	4050 / Jun-28-99	4360 / Sep-30-99	5112 / Dec-29-99
South America	600 / Feb-27-99	490 / Jun-03-99	315 / Aug-21-99	580 / Nov-30-99

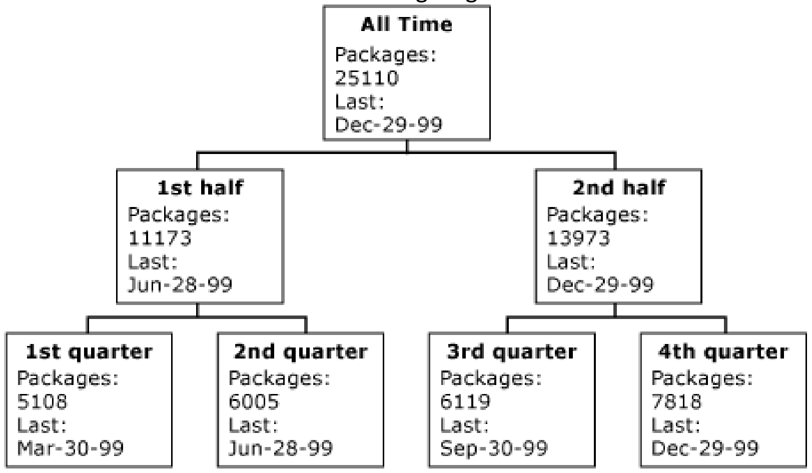
The smaller alphanumeric values around the cube are the members of the dimensions. Example members are ground (member of the Route dimension), Africa (member of the Source dimension), and 1st quarter (member of the Time dimension).

Measures
The values within the cube cells represent the two measures, Packages and Last. The Packages measure represents the number of imported packages, and the **Sum** function is used to aggregate the facts. The Last measure represents the date of receipt, and the **Max** function is used to aggregate the facts.

Dimensions
The Route dimension represents the means by which the imports reach their destination. Members of this dimension include ground, nonground, air, sea, road, or rail. The Source dimension represents the locations where the imports are produced, such as Africa or Asia. The Time dimension represents the quarters and halves of a single year.

Aggregates
Business users of a cube can determine the value of any measure for each member of every dimension, regardless of the level of the member within the dimension, because Analysis Services aggregates values at upper levels as needed. For example, the measure values in

the preceding illustration can be aggregated according to a standard calendar hierarchy by using the Calendar Time hierachy in the Time dimension as illustrated in the following diagram.



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

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

After a cube is defined, you can create new aggregations, or you can change existing aggregations to set options such as whether aggregations are precalculated during processing or calculated at query time. **Related topic:** Aggregations and Aggregation Designs (SSAS).

Mapping Measures, Attributes, and Hierarchies

The measures, attributes, and hierarchies in the example cube are derived from the following columns in the cube's fact and dimension tables.

Measure or attribute (level)	Members	Source table	Source column	Sample column value
Packages measure	Not applicable	ImportsFactTable	Packages	12
Last measure	Not applicable	ImportsFactTable	Last	May-03-99
Route Category level in Route dimension	nonground,ground	RouteDimensionTable	Route_Category	Nonground
Route attribute in Route dimension	air,sea,road,rail	RouteDimensionTable	Route	Sea
Hemisphere attribute in Source dimension	Eastern Hemisphere,Western Hemisphere	SourceDimensionTable	Hemisphere	Eastern Hemisphere
Continent attribute in Source dimension	Africa,Asia,AustraliaEurope,N. America,S. America	SourceDimensionTable	Continent	Europe
Half attribute in Time dimension	1st half,2nd half	TimeDimensionTabl	Half	2nd half

		e		
Quarter attribute in Time dimension	1st quarter,2nd quarter,3rd quarter,4th quarter	TimeDimensionTable	Quarter	3rd quarter

Data in a single cube cell is usually derived from multiple rows in the fact table. For example, the cube cell at the intersection of the air member, the Africa member, and the 1st quarter member contains a value that is derived by aggregating the following rows in the **ImportsFactTable** fact table.

Import_ReceiptKey	RouteKey	SourceKey	TimeKey	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, each row has the same values for the **RouteKey**, **SourceKey**, and **TimeKey** columns, indicating that these rows contribute to the same cube cell.
The example shown here represents a very simple cube, in that the cube has a single measure group, and all the dimension tables are joined to the fact table in a star schema. Another common schema is a snowflake schema, in which one or more dimension tables join to another dimension table, rather than joining directly to the fact table. **Related topic:** Dimensions (Analysis Services).
The example shown here contains only a single fact table. When a cube has multiple fact tables, the measures from each fact table are organized into measure groups, and a measure group is related to a specific set of dimensions by defined dimension relationships. These relationships are defined by specifying the participating tables in the data source view and the granularity of the relationship.

Pasted from <http://msdn.microsoft.com/en-us/library/ms174587.aspx>

Measures and Measure Groups

Sunday, August 03, 2008 2:33 AM

Measures and Measure Groups

A measure represents a column that contains quantifiable data, usually numeric, that can be aggregated. A measure is generall y mapped to a column in a fact table.

You can also use a *measure expression* to define the value of a measure, based on a column in a fact table as modified by a Multidimensional Expression. A measure expression enables weighting of measure values; for example, currency conversion can be used to weight a sales measur e by an exchange rate.

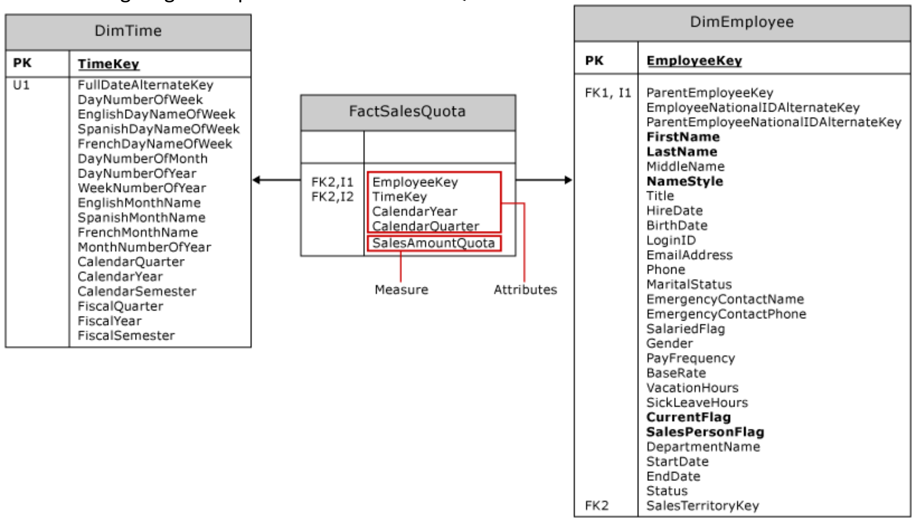
Attribute columns from dimension tables can be used to define measures, but such measures are typically semiadditive or nonad ditive in terms of their aggregation behavior.

You can also define a measure as a *calculated member* by using a Multidimensional Expressions (MDX) to provide a calculated value for a measure based on other measures in the cube. Calculated members add flexibility and analysis capability to a cube in Analysis Services. .

Measure Groups

In a cube, measures are grouped by their underlying fact tables into measure groups. Measure groups are used to associate dim ensions with measures. Measure groups are also used for measures that have distinct count as their aggregation behavior. Placing each distinct count measure into its own measure group optimizes aggregation processing.

The following diagram represents the **FactSalesQuota** fact table and the two dimension tables associated with it, **DimTime** and **DimEmployee**.



The **FactSalesQuota** fact table defines the Sales Quotas measure group of the Adventure Works cube, and the **DimTime** and **DimEmployee** dimension tables define the Time and Employee dimensions in the Adventure Works DW sample Analysis Services project.

The fact table contains two basic types of columns: attribute columns and measure columns. Attribute columns are used to crea te foreign key relationships to dimension tables, so that the quantifiable data in the measure columns can be organized by the data contained in the dimen sion tables. Attribute columns are also used to define the granularity of a fact table and its measure group. Measure columns define the measures co ntained by a measure group. In the **FactSalesQuota** fact table, the **SalesAmountQuota** column is used to define the Sales Amount Quota measure. This measure is contained by the Sales Quotas measure group, and is organized by the Time and Employee dimensions.

Granularity

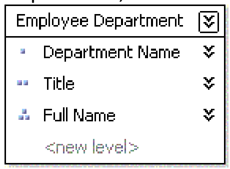
Granularity refers to the level of detail supported by a fact table. For example, the **FactSalesQuota** fact table has a foreign key relationship with the **DimEmployee** table, on the **EmployeeKey** primary key column. In other words, each record in the **FactSalesQuota** table is related to a single employee; therefore, the granularity of the measure group as viewed from the Employee dimension is at the individual employee level.

The granularity of a measure group can never be set finer than the lowest level of the dimension from which the measure group is viewed, but the granularity can be made coarser by using additional attributes. For example, the **FactSalesQuota** fact table uses three columns, **TimeKey**, **CalendarYear**, and **CalendarQuarter**, to establish the granularity of the relationship with the **DimTime** table. As a result, the granularity of the measure group as viewed from the Time dimension is by calendar quarter, and not by day, which is the lowest level of the Time dimension.

You can specify the granularity of a measure group with relation to a specific dimension by using the **Dimension Usage** tab of the Cube Designer.

Aggregate Functions

When a dimension is used to organize measures in a measure group, the measure is summarized along the hierarchies contained i n that dimension. The summation behavior depends on the aggregate function specified for the measure. For example, the Employee dimension has a hie rarchy named Employee Department, which is structured in levels like those in the following diagram:



Each record in the **FactSalesQuota** fact table is directly related to a single record in the **DimEmployee** dimension table by a foreign key relationship with the **EmployeeKey** column. Therefore, each value in the Sales Amount Quota measure relates to a single leaf member in the Full Name level of the Employee

Department hierarchy, and can be directly loaded from the fact table. However, the value of the Sales Amount Quota measure fo r members in levels above Full Name cannot be directly loaded, because each member represents more than one record. In other words, a single member fro m the Title level may have several employees associated with it, and therefore several members in the Full Name level. The measure values for these nonleaf members are not directly loaded from the data source, but are instead aggregated from the members below it in the hierarchy.
Not all measures are derived directly from a value stored in a column of the fact table. For example, the Sales Person Count measure defined in the Sales Quota measure group of the Adventure Works cube in the Adventure Works DW sample Analysis Services project is actually based on the count of unique values (or distinct count) in the **EmployeeKey** column of the **FactSalesQuota** fact table.
The aggregation behavior of each measure is determined by the aggregation function associated with the measure.

Dimension Relationships

Sunday, August 03, 2008 2:30 AM

Dimension Relationships

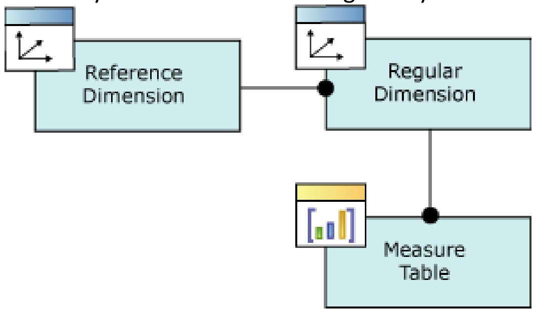
Dimension usage defines the relationships between a cube dimension and the measure groups in a cube. A cube dimension is an instance of a database dimension that is used in a specific cube. A cube can, and frequently does, have cube dimensions that are not directly related to a measure group, but which might be indirectly related to the measure group through another dimension or measure group. When you add a database dimension or measure group to a cube, Microsoft SQL Server 2005 Analysis Services (SSAS) tries to determine dimension usage by examining relationships between the dimension tables and fact tables in the cube's data source view, and by examining the relationships between attributes in dimensions. Analysis Services automatically sets the dimension usage settings for the relationships that it can detect.
A relationship between a dimension and a measure group consists of the dimension and fact tables participating in the relationship and a granularity attribute that specifies the granularity of the dimension in the particular measure group.

Regular Dimension Relationships

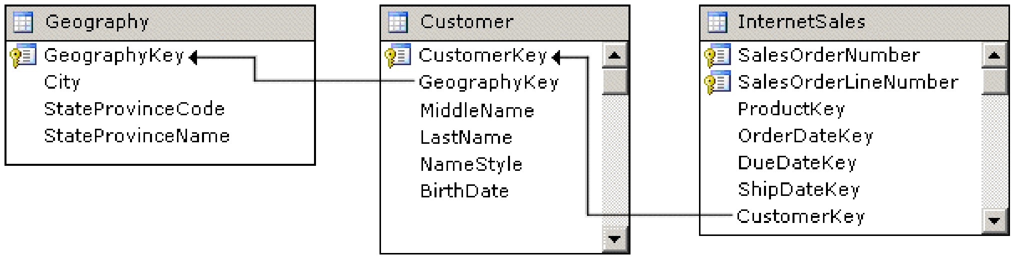
A regular dimension relationship between a cube dimension and a measure group exists when the key column for the dimension is joined directly to the fact table. This direct relationship is based on a primary key–foreign key relationship in the underlying relational database, but might also be based on a logical relationship that is defined in the data source view. A regular dimension relationship represents the relationship between dimension tables and a fact table in a traditional star schema design.

Reference Dimension Relationships

A reference dimension relationship between a cube dimension and a measure group exists when the key column for the dimension is joined indirectly to the fact table through a key in another dimension table, as shown in the following illustration.



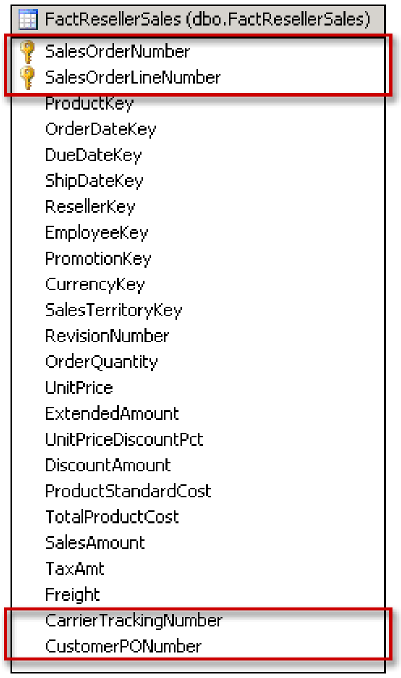
A reference dimension relationship represents the relationship between dimension tables and a fact table in a snowflake schema design. When dimension tables are connected in a snowflake schema, you can define a single dimension using columns from multiple tables, or you can define separate dimensions based on the separate dimension tables and then define a link between them using the reference dimension relationship setting. The following figure shows one fact table named **InternetSales**, and two dimension tables called **Customer** and **Geography**, in a snowflake schema.



You can create a dimension with the **Customer** table as the dimension main table and the **Geography** table included as a related table. A regular relationship is then defined between the dimension and the InternetSales measure group.
Alternatively, you can create two dimensions related to the InternetSales measure group: a dimension based on the **Customer** table, and a dimension based on the **Geography** table. You can then relate the Geography dimension to the InternetSales measure group using a reference dimension relationship using the Customer dimension. In this case, when the facts in the InternetSales measure group are dimensioned by the Geography dimension, the facts are dimensioned by customer and by geography. If the cube contained a second measure group named Reseller Sales, you would be unable to dimension the facts in the Reseller Sales measure group by Geography because no relationship would exist between Reseller Sales and Geography.
There is no limit to the number of reference dimensions that can be chained together, as shown in the following illustration.

Fact Dimension Relationships

Fact dimensions, frequently referred to as degenerate dimensions, are standard dimensions that are constructed from attribute columns in fact tables instead of from attribute columns in dimension tables. Useful dimensional data is sometimes stored in a fact table to reduce duplication. For example, the following diagram displays the **FactResellerSales** fact table, from the **Adventure Works DW** sample database.


FactResellerSales (dbo.FactResellerSales)
SalesOrderNumber
SalesOrderLineNumber
ProductKey
OrderDateKey
DueDateKey
ShipDateKey
ResellerKey
EmployeeKey
PromotionKey
CurrencyKey
SalesTerritoryKey
RevisionNumber
OrderQuantity
UnitPrice
ExtendedAmount
UnitPriceDiscountPct
DiscountAmount
ProductStandardCost
TotalProductCost
SalesAmount
TaxAmt
Freight
CarrierTrackingNumber
CustomerPONumber

The table contains attribute information not only for each line of an order issued by a reseller, but about the order itself. The attributes circled in the previous diagram identify the information in the **FactResellerSales** table that could be used as attributes in a dimension. In this case, two additional pieces of information, the carrier tracking number and the purchase order number issued by the reseller, are represented by the CarrierTrackingNumber and CustomerPONumber attribute columns. This information is interesting—for example, users would definitely be interested in seeing aggregated information, such as the total product cost, for all the orders being shipped under a single tracking number. But, without a dimension data for these two attributes cannot be organized or aggregated.
In theory, you could create a dimension table that uses the same key information as the FactResellerSales table and move the other two attribute columns, CarrierTrackingNumber and CustomerPONumber, to that dimension table. However, you would be duplicating a significant portion of data and adding unnecessary complexity to the data warehouse to represent just two attributes as a separate dimension.

Note:

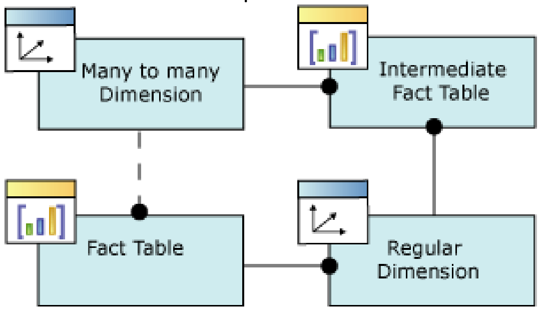
Fact dimensions are frequently used to support drillthrough actions.

Note:

Fact dimensions must be incrementally updated after every update to the measure group that is referenced by the fact relationship. If the fact dimension is a ROLAP dimension, the Analysis Services processing engine drops any caches and incrementally processes the measure group.

Many to Many Dimension Relationships

In most dimensions, each fact joins to one and only one dimension member, and a single dimension member can be associated with multiple facts. In relational database terminology, this is referred to as a one-to-many relationship. However, it is frequently useful to join a single fact to multiple dimension members. For example, a bank customer might have multiple accounts (checking, saving, credit card, and investment accounts), and an account can also have joint or multiple owners. The Customer dimension constructed from such relationships would then have multiple members that relate to a single account transaction.



SQL Server 2005 Analysis Services (SSAS) lets you define a many-to-many relationship between a dimension and a fact table.

Note:

To support a many-to-many dimension relationship, the data source view must have established a foreign key relationship between all the tables involved, as shown in the previous diagram. Otherwise, you will be unable to select the correct intermediate measure group when establishing the relationship in the **Dimension Usage** tab of Dimension Designer.

Calculations

Sunday, August 03, 2008 2:32 AM

Calculations
A calculation is a Multidimensional Expressions (MDX) expression or script that is used to define a calculated member, a named set, or a scoped assignment in a cube in Microsoft SQL Server 2005 Analysis Services. Calculations let you add objects that are defined not by the data of the cube, but by expressions that can reference other parts of the cube, other cubes, or even information outside the Analysis Services database. Calculations let you extend the capabilities of a cube, adding flexibility and power to business intelligence applications.

Calculated Members
A calculated member is a member whose value is calculated at run time using a Multidimensional Expressions (MDX) expression that you specify when you define the calculated member. A calculated member is available to business intelligence applications just like any other member. Calculated members do not increase the size of the cube because only the definitions are stored in the cube; values are calculated in memory as required to answer a query.
Calculated members can be defined for any dimension, including the measures dimension. Calculated members created on the Measures dimension are called calculated measures.
Although calculated members are typically based on data that already exists in the cube, you can create complex expressions by combining data with arithmetic operators, numbers, and functions. You can also use MDX functions, such as LookupCube, to access data in other cubes in the Analysis Services database. Analysis Services includes standardized Visual Studio function libraries, and you can use stored procedures to retrieve data from sources other than the current Analysis Services database.
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 by combining the existing measures in the following expression:
Copy Code
```
([Measures].[Price_to_Ship] - [Measures].[Cost_to_Ship]) /
[Measures].[Volume_in_Cubic_Meters]
```

After you create the calculated member, the Profit_per_Cubic_Meter appears together with the other measures the next time that the Shipments cube is browsed.
To create calculated members, use the **Calculation**s tab in Cube Designer.

Named Sets
A named set is a CREATE SET MDX statement expression that returns a set. The MDX expression is saved as part of the definition of a cube in Microsoft SQL Server 2005 Analysis Services (SSAS). A named set is created for reuse in Multidimensional Expressions (MDX) queries. A named set enables business users to simplify queries, and use a set name instead of a set expression for complex, frequently used set expressions.

Script Commands
A script command is an MDX script, included as part of the definition of the cube. Script commands let you perform almost any action that is supported by MDX on a cube, such as scoping a calculation to apply to only part of the cube. In SQL Server 2005 Analysis Services (SSAS), MDX scripts can apply either to the whole cube or to specific sections of the cube, at specific points throughout the execution of the script. The default script command, which is the CALCULATE statement, populates cells in the cube with aggregated data based on the default scope.
The default scope is the whole cube, but you can define a more limited scope, known as a subcube, and then apply an MDX script to only that particular cube space. The SCOPE statement defines the scope of all subsequent MDX expressions and statements in the calculation script until the scope is terminated or redefined. The THIS statement is then used to apply an MDX expression to the current scope. You can use the BACK_COLOR statement to specify a background cell color for the cells in the current scope, to help you during debugging.
For example, you can use a script command to allocate sales quotas to employees across time and sales territory based on the weighted values of sales for a prior time period.

Key Performance Indicators (KPIs)

Sunday, August 03, 2008 2:33 AM

Key Performance Indicators

In business terminology, a Key Performance Indicator (KPI) is a quantifiable measurement for gauging business success. A KPI is frequently evaluated over time. For example, the sales department of an organization may use monthly gross profit as a KPI, but the human resources department of the same organization may use quarterly employee turnover. Each is an example of a KPI. Business executives frequently consume KPIs that are grouped together in a business scorecard to obtain a quick and accurate historical summary of business success.

In Microsoft SQL Server 2005 Analysis Services (SSAS), a KPI is a collection of calculations, which are associated with a measure group in a cube, that are used to evaluate business success. Typically, these calculations are a combination of Multidimensional Expressions (MDX) expressions and calculated members. KPIs also have additional metadata that provides information about how client applications should display the results of a KPI's calculation.

One key advantage of KPIs in Analysis Services is that they are server-based KPIs that are consumable by different client applications. A server-based KPI presents a single version of the truth, compared to separate versions of the truth from separate client applications. Moreover, performing the sometimes complex calculations on the server instead of on each client computer may have performance benefits.

Common KPI Terms

The following table provides definitions for common KPI terms in Analysis Services.

Term	Definition
Goal	An MDX numeric expression or a calculation that returns the target value of the KPI.
Value	An MDX numeric expression that returns the actual value of the KPI.
Status	An MDX expression that represents the state of the KPI at a specified point in time. The status MDX expression should return a normalized value between -1 and 1. Values equal to or less than -1 will be interpreted as "bad" or "low." A value of zero (0) is interpreted as "acceptable" or "medium." Values equal to or greater than 1 will be interpreted as "good" or "high." An unlimited number of intermediate values can optionally be returned and can be used to display any number of additional states, if supported by the client application.
Trend	An MDX expression that evaluates the value of the KPI over time. The trend can be any time-based criterion that is useful in a specific business context. The trend MDX expression enables a business user to determine whether the KPI is improving over time or degrading over time.
Status indicator	A visual element that provides a quick indication of the status for a KPI. The display of the element is determined by the value of the MDX expression that evaluates status.
Trend indicator	A visual element that provides a quick indication of the trend for a KPI. The display of the element is determined by the value of the MDX expression that evaluates trend.
Display folder	The folder in which the KPI will appear when a user is browsing the cube.
Parent KPI	A reference to an existing KPI that uses the value of the child KPI as part of computation of the parent KPI. Sometimes, a single KPI will be a computation that consists of the values for other KPIs. This property facilitates the correct display of the child KPIs underneath the parent KPI in client applications.
Current time member	An MDX expression that returns the member that identifies the temporal context of the KPI.
Weight	An MDX numeric expression that assigns a relative importance to a KPI. If the KPI is assigned to a parent KPI, the weight is used to proportionally adjust the results of the child KPI value when calculating the value of the parent KPI.

Parent KPIs

An organization may track different business metrics at different levels. For example, only two or three KPIs may be used to gauge business success for the whole company, but these company-wide KPIs may be based on three or four other KPIs tracked by the business units throughout the company. Also, business units in a company may use different statistics to calculate the same KPI, the results of which are rolled up to the company-wide KPI.

Analysis Services lets you define a parent-child relationship between KPIs. This parent-child relationship lets the results of the child KPI be used to calculate the results of the parent KPI. Client applications can also use this relationship to appropriately display parent and child KPIs.

Weights

Weights can also be assigned to child KPIs. Weights enable Analysis Services to proportionally adjust the results of the child KPI when calculating the value of the parent KPI.

Retrieving and Displaying KPIs

The display of KPIs depends on the implementation of the client application. For example, selecting **Browser View** on the toolbar on the **KPIs** tab of Cube Designer demonstrates one possible client implementation, with graphics used to display status and trend indicators, display folders used to group KPIs, and child KPIs displayed under parent KPIs.

You can use MDX functions to retrieve individual sections of the KPI, such as the value or goal, for use in MDX expressions, statements, and scripts.

Actions

Sunday, August 03, 2008 2:35 AM

Actions
In Microsoft SQL Server 2005 Analysis Services, an action is a stored MDX statement that can be presented to and employed by client applications. In other words, an action is a client command that is defined and stored on the server. An action also contains information that specifies when and how the MDX statement should be displayed and handled by the client application. The operation that is specified by the action can start an application, using the information in the action as a parameter, or can retrieve information based on criteria supplied by the action.
Actions enable business users to act upon the outcomes of their analyses. By saving and reusing actions, end users can go beyond traditional analysis, which typically ends with presentation of data, and initiate solutions to discovered problems and deficiencies, thereby extending the business intelligence application beyond the cube. Actions can transform the client application from a sophisticated data rendering tool into an integral part of the enterprise's operational system. Instead of focusing on sending data as input to operational applications, end users can "close the loop" on the decision-making process. This ability to transform analytical data into decisions is crucial to the successful business intelligence application.
For example, a business user browsing a cube notices that the current stock of a certain product is low. The client application provides to the business user a list of actions, all related to low product stock value, that are retrieved from the Analysis Services database, The business user selects the Order action for the member of the cube that represents the product. The Order action initiates a new order by calling a stored procedure in the operational database. This stored procedure generates the appropriate information to send to the order entry system.
You can exercise flexibility when you create actions: for example, an action can launch an application, or retrieve information from a database. You can configure an action to be triggered from almost any part of a cube, including dimensions, levels, members, and cells, or create multiple actions for the same portion of a cube. You can also pass string parameters to the launched applications and specify the captions displayed to end users as the action runs.

Important:

In order for a business user to use actions, the client application employed by the business user must support actions.

Types of Actions
The following table lists the types of actions that are included in Analysis Services:

Action Type	Description
CommandLine	Executes a command at the command prompt
Dataset	Returns a dataset to a client application.
Drillthrough	Returns a drillthrough statement as an expression, which the client executes to return a rowset
Html	Executes an HTML script in an Internet browser
Proprietary	Performs an operation by using an interface other than those listed in this table.
Report	Submits a parameterized URL-based request to a report server and returns a report to a client application.
Rowset	Returns a rowset to a client application.

Statement	Runs an OLE DB command.
URL	Displays a dynamic Web page in an Internet browser.

Resolving and Executing Actions

When a business user accesses the object for which the command object is defined, the statement associated with the action is automatically resolved, which makes it available to the client application, but the action is not automatically executed. The action is executed only when the business user performs the client-specific operation that initiates the action. For example, a client applications might display a list of actions as a pop-up menu when the business user right-clicks on a particular member or cell.

Partitions

Sunday, August 03, 2008 2:35 AM

Partitions (Analysis Services)

Updated: **14 April 2006**

Partitions are used by Microsoft SQL Server 2005 Analysis Services to manage and store data and aggregations for a measure group in a cube. Every measure group has at least one partition; this partition is created when the measure group is defined. When you create a new partition for a measure group, the new partition is added to the set of partitions that already exist for the measure group. The measure group reflects the combined data that is contained in all its partitions. This means that you must ensure that the data for a partition in a measure group is exclusive of the data for any other partition in the measure group to ensure that data is not reflected in the measure group more than once. The original partition for a measure group is based on a single fact table in the data source view of the cube. When there are multiple partitions for a measure group, each partition can reference a different table in either the data source view or in the underlying relational data source for the cube. More than one partition in a measure group can reference the same table, if each partition is restricted to different rows in the table.

Partitions are a powerful and flexible means of managing cubes, especially large cubes. For example, a cube that contains sales information can contain a partition for the data of each past year and also partitions for each quarter of the current year. Only the current quarter partition needs to be processed when current information is added to the cube; processing a smaller amount of data will improve processing performance by decreasing processing time. At the end of the year the four quarterly partitions can be merged into a single partition for the year and a new partition created for the first quarter of the new year. Further, this new partition creation process can be automated as part of your data warehouse loading and cube processing procedures.

Partitions are not visible to business users of the cube. However, administrators can configure, add, or drop partitions. Each partition is stored in a separate set of files. The aggregate data of each partition can be stored on the instance of Analysis Services where the partition is defined, on another instance of Analysis Services, or in the data source that is used to supply the partition's source data. Partitions allow the source data and aggregate data of a cube to be distributed across multiple hard drives and among multiple server computers. For a cube of moderate to large size, partitions can greatly improve query performance, load performance, and ease of cube maintenance.

The storage mode of each partition can be configured independently of other partitions in the measure group. Partitions can be stored by using any combination of options for source data location, storage mode, proactive caching, and aggregation design. Options for real-time OLAP and proactive caching let you balance query speed against latency when you design a partition. Storage options can also be applied to related dimensions and to facts in a measure group. This flexibility lets you design cube storage strategies appropriate to your needs.

Partition Structure

The structure of a partition must match the structure of its measure group, which means that the measures that define the measure group must also be defined in the partition, along with all related dimensions. Therefore, when a partition is created, it automatically inherits the same set of measures and related dimensions that were defined for the measure group.

However, each partition in a measure group can have a different fact table, and these fact tables can be from different data sources. When different partitions in a measure group have different fact tables, the tables must be sufficiently similar to maintain the structure of the measure group, which means that the processing query returns the same columns and same data types for all fact tables for all partitions.

When fact tables for different partitions are from different data sources, the source tables for any related dimensions, and also any intermediate fact tables, must also be present in all data sources and must have the same structure in all the databases. Also, all dimension table columns that are used to define attributes for cube dimensions related to the measure group must be present in all of the data sources. There is no need to define all the joins between the source table of a partition and a related dimension table if the partition source table has the identical structure as the source table for the measure group.

Columns that are not used to define measures in the measure group can be present in some fact tables but absent in others. Similarly, columns that are not used to define attributes in related dimension tables can be present in some databases but absent in others. Tables that are not used for either fact tables or related dimension tables can be present in some databases but absent in others.

Data Sources and Partition Storage

A partition is based either on a table or view in a data source, or on a table or named query in a data source view. The location where partition data is stored is defined by the data source binding. Typically, you can partition a measure group horizontally or vertically:

- In a horizontally partitioned measure group, each partition in a measure group is based on a separate table. This kind of partitioning is appropriate when data is separated into multiple tables. For example, some relational databases have a separate table for each month's data.
- In a vertically partitioned measure group, a measure group is based on a single table, and each partition is based on a source

system query that filters the data for the partition. For example, if a single table contains several months data, the measure group could still be partitioned by month by applying a Transact SQL WHERE clause that returns a separate month's data for each partition.

Each partition has storage settings that determine whether the data and aggregations for the partition are stored in the local instance of Analysis Services or in a remote partition using another instance of Analysis Services. The storage settings can also specify the storage mode and whether proactive caching is used to control latency for a partition.

Incremental Updates

When you create and manage partitions in multiple-partition measure groups, you must take special precautions to guarantee that cube data is accurate. Although these precautions do not usually apply to single-partition measure groups, they do apply when you incrementally update partitions. When you incrementally update a partition, a new temporary partition is created that has a structure identical to that of the source partition. The temporary partition is processed and then merged with the source partition. Therefore, you must ensure that the processing query that populates the temporary partition does not duplicate any data already present in an existing partition.

Perspectives

Sunday, August 03, 2008 2:35 AM

Perspectives

Cubes can be very complex objects for users to explore in Microsoft SQL Server 2005 Analysis Services. A single cube can represent the contents of a complete data warehouse, with multiple measure groups in a cube representing multiple fact tables, and multiple dimensions based on multiple dimension tables. Such a cube can be very complex and powerful, but daunting to users who may only need to interact with a small part of the cube in order to satisfy their business intelligence and reporting requirements.

In Microsoft SQL Server 2005 Analysis Services (SSAS), you can use a perspective to reduce the perceived complexity of a cube in Analysis Services. A perspective defines a viewable subset of a cube that provides focused, business-specific or application-specific viewpoints on the cube. The perspective controls the visibility of objects that are contained by a cube. The following objects can be displayed or hidden in a perspective:

- Dimensions
- Attributes
- Hierarchies
- Measure groups
- Measures
- Key Performance Indicators (KPIs)
- Calculations (calculated members, named sets, and script commands)
- Actions

For example, the **Adventure Works** cube in the **Adventure Works DW** sample Analysis Services database contains eleven measure groups and twenty-one different cube dimensions, representing sales, sales forecasting, and financial data. A client application can directly reference the complete cube, but this viewpoint may be overwhelming to a user trying to extract basic sales forecasting information. Instead, the same user can use the **Sales Targets** perspective to limit the view of the **Adventure Works** cube to only those objects relevant to sales forecasting.

Objects in a cube that are not visible to the user through a perspective can still be directly referenced and retrieved using XML for Analysis (XMLA), Multidimensional Expressions (MDX), or Data Mining Extensions (DMX) statements. Perspectives do not restrict access to objects in a cube and should not be used as such; instead, perspectives are used to provide a better user experience while accessing a cube. A perspective is a read-only view of the cube; objects in the cube cannot be renamed or changed by using a perspective. Similarly, the behavior or features of a cube, such as the use of visual totals, cannot be changed by using a perspective.

Security

Perspectives are not meant to be used as a security mechanism, but as a tool for providing a better user experience in business intelligence applications. All security for a particular perspective is inherited from the underlying cube. For example, perspectives cannot provide access to objects in a cube to which a user does not already have access. - Security for the cube must be resolved before access to objects in the cube can be provided through a perspective.

Cube Translations

Sunday, August 03, 2008 2:36 AM

Cube Translations

In Microsoft SQL Server 2005 Analysis Services (SSAS), a cube translation is a language-specific representation of the name of a cube object, such as a caption or a display folder. Analysis Services also supports translations of dimension and member names.

Translations provide server support for client applications that can support multiple languages. Frequently, users from different countries view cube data. It is useful to be able to translate various elements of a cube into a different language so that these users can view and understand the cube's metadata. For example, a business user in France can access a cube from a workstation with a French locale setting, and view the object property values in French. Similarly, a business user in Germany can access the same cube from a workstation with a German locale setting and view the object property values in German.

The collation and language information for the client computer is stored in the form of a locale identifier (LCID). Upon connection, the client passes the LCID to the instance of Analysis Services. The instance uses the LCID to determine which set of translations to use when providing metadata for Analysis Services objects to each business user. If an Analysis Services object does not contain the specified translation, the default language is used to return the content back to the client.

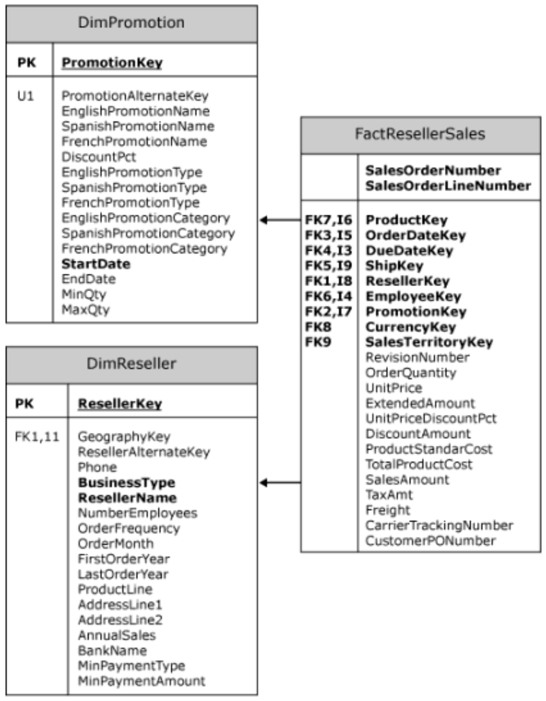
Dimensions

Sunday, August 03, 2008 2:40 AM

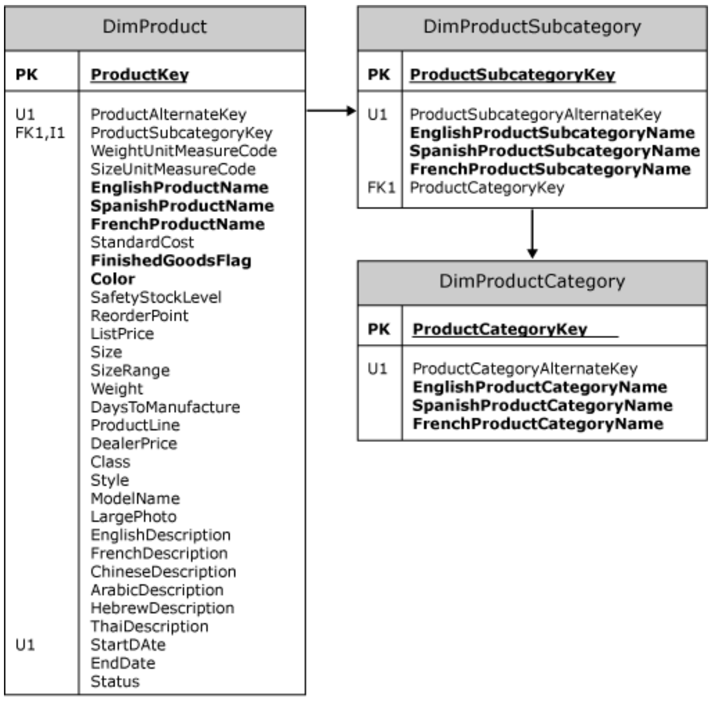
Dimensions (Analysis Services)
In Microsoft SQL Server 2005 Analysis Services (SSAS), dimensions are a fundamental component of cubes. Dimensions organize data with relation to an area of interest, such as customers, stores, or employees, to users. Dimensions in Analysis Services contain attributes that correspond to columns in dimension tables. These attributes appear as attribute hierarchies and can be organized into user-defined hierarchies, or can be defined as parent-child hierarchies based on columns in the underlying dimension table. Hierarchies are used to organize measures that are contained in a cube. The following topics provide an overview of dimensions, attributes, and hierarchies.

Introduction to Dimensions
All Microsoft SQL Server 2005 Analysis Services dimensions are groups of attributes based on columns from tables or views in a data source view. Dimensions exist independent of a cube, can be used in multiple cubes, can be used multiple times in a single cube, and can be linked between Analysis Services.instances. A dimension that exists independent of a cube is called a database dimension and an instance of a database dimension within a cube is called a cube dimension.

Dimension based on a Star Schema Design
The structure of a dimension is largely driven by the structure of the underlying dimension table or tables. The simplest structure is called a star schema, where each dimension is based on a single dimension table that is directly linked to the fact table by a primary key - foreign key relationship.
The following diagram illustrates a subsection of the **AdventureWorksDW** sample database, in which the **FactResellerSales** fact table is related to two dimension tables, **DimReseller** and **DimPromotion**. The **ResellerKey** column in the **FactResellerSales** fact table defines a foreign key relationship to the **ResellerKey** primary key column in the **DimReseller** dimension table. Similarly, the **PromotionKey** column in the **FactResellerSales** fact table defines a foreign key relationship to the **PromotionKey** primary key column in the **DimPromotion** dimension table.



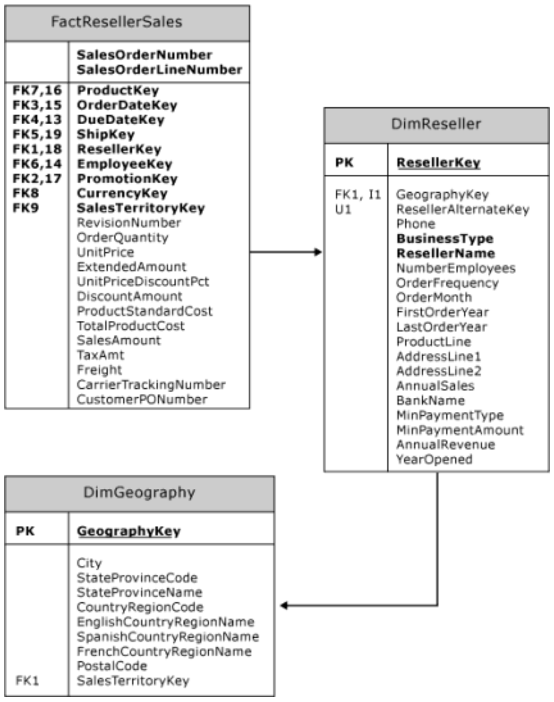
Dimension based on a Snowflake Schema Design
Frequently, a more complex structure is required because information from multiple tables is required to define the dimension. In this structure, called a snowflake schema, each dimension is based on attributes from columns in multiple tables linked to each other and ultimately to the fact table by primary key - foreign key relationships. For example, the following diagram illustrates the tables required to completely describe the Product dimension in the **AdventureWorksDW** sample project:



To completely describe a product, the product's category and subcategory must be included in the Product dimension. However, that information does not reside directly in the main table for the **DimProduct** dimension. A foreign key relationship from **DimProduct** to **DimProductSubcategory**, which in turn has a foreign key relationship to the **DimProductCategory** table, makes it possible to include the information for product categories and subcategories in the Product dimension.

Snowflake Schema versus Reference Relationship

Sometimes, you may have a choice between using a snowflake schema to define attributes in a dimension from multiple tables, or defining two separate dimensions and defining a reference dimension relationship between them. The following diagram illustrates such a scenario.



In the previous diagram, the **FactResellerSales** fact table does not have a foreign key relationship with the **DimGeography** dimension table. However, the **FactResellerSales** fact table does have a foreign key relationship with the **DimReseller** dimension table, which in turn has a foreign key relationship with the **DimGeography** dimension table. To define a Reseller dimension that contains geography information about each reseller, you would have to retrieve these attributes from the **DimGeography** and the **DimReseller** dimension tables. However, in Analysis Services, you can achieve the same result by creating two separate dimensions and linking them in a measure group by defining a reference dimension relationship between the two dimensions.

One advantage of using reference dimension relationships in this scenario is that you could create a single geography dimension and then create multiple cube dimensions based on the geography dimension, without requiring any additional storage space. For example, you could link one of the geography cube dimensions to a reseller dimension and another of the geography cube dimensions to a customer dimension.

Processing a Dimension

After you create a dimension, you must process the dimension before you can view the members of the attributes and hierarchies in the dimension. After the structure of a dimension is changed or the information in its underlying tables is updated, you have to process the dimension again before you can view the changes. When you process a dimension after structural changes, you must also process any cubes that include the dimension - or the cube will not be viewable.

Security

All the subordinate objects of a dimension, including hierarchies, levels, and members, are secured using roles in Analysis Services. Dimension security can be applied for all the cubes in the database that use the dimension, or for only a specific cube.

Attributes and Attribute Hiearchies

Sunday, August 03, 2008 2:43 AM

Attributes and Attribute Hierarchies
Dimensions are collections of attributes, which are bound to one or more columns in a table or view in the data source view.

Key Attribute
Each dimension contains a key attribute. Each attribute bound to one or more columns in a dimension table. The key attribute is the attribute in a dimension that identifies the columns in the dimension main table that are used in foreign key relationships to the fact table. Typically, the key attribute represents the primary key column or columns in the dimension table. You can define a logical primary key on a table in a data source view which has no physical primary key in the underlying data source. When defining key attributes, the Cube Wizard and Dimension Wizard try to use the primary key columns of the dimension table in the data source view. If the dimension table does not have a logical primary key or physical primary key defined, the wizards may not be able to correctly define the key attributes for the dimension.

Binding an Attribute to Columns in Data Source View Tables or Views
An attribute is bound to columns in one or more data source view tables or views. An attribute is always bound to one or more key columns, which determines the members that are contained by the attribute. By default, this is the only column to which an attribute is bound. An attribute can also be bound to one or more additional columns for specific purposes. For example, an attribute's **NameColumn** property determines the name that appears to the user for each attribute member - this property of the attribute can be bound to a particular dimension column through a data source view or can be bound to a calculated column in the data source view.

Attribute Hierarchies
By default, attribute members are organized into two level hierarchies, consisting of a leaf level and an All level. The All level contains the aggregated value of the attribute's members across the measures in each measure group to which the dimension of which the attribute is related is a member. However, if the **IsAggregatable** property is set to False, the All level is not created.
Attributes can be, and typically are, arranged into user-defined hierarchies that provide the drill-down paths by which users can browse the data in the measure groups to which the attribute is related. In client applications, attributes can be used to provide grouping and constraint information. When attributes are arranged into user-defined hierarchies, you define relationships between hierarchy levels when levels are related in a many-to-one or a one-to-one relationship (called a *natural* relationship). For example, in a Calendar Time hierarchy, a Day level should be related to the Month level, the Month level related to the Quarter level, and so on. Defining relationships between levels in a user-defined hierarchy enables Analysis Services to define more useful aggregations to increase query performance and can also save memory during processing performance, which can be important with large or complex cubes.

Attribute Relationships, Star Schemas, and Snowflake Schemas
By default, in a star schema, all attributes are directly related to the key attribute, which enables users to browse the facts in the cube based on any attribute hierarchy in the dimension. In a snowflake schema, an attribute is either directly linked to the key attribute if their underlying table is directly linked to the fact table or is indirectly linked by means of the attribute that is bound to the key in the underlying table that links the snowflake table to the directly linked table.

User-Defined Hierarchies

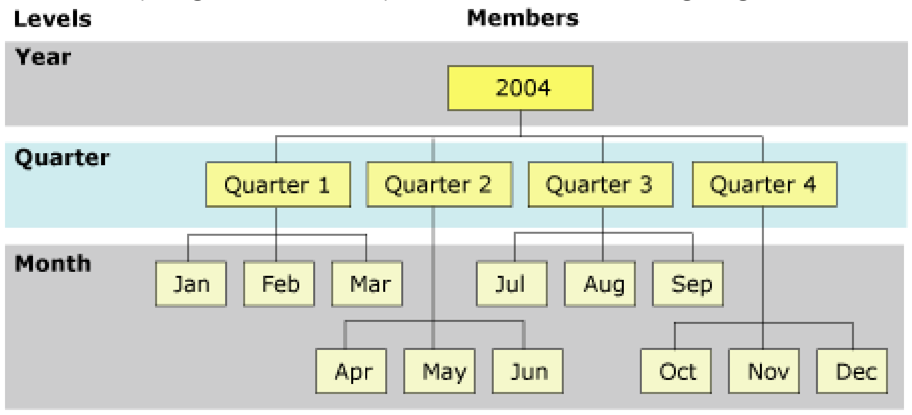
Sunday, August 03, 2008 2:44 AM

User-Defined Hierarchies

User-defined hierarchies are user-defined hierarchies of attributes that are used in Microsoft SQL Server 2005 Analysis Services to organize the members of a dimension into hierarchical structures and provide navigation paths in a cube. For example, the following table defines a dimension table for a time dimension. The dimension table supports three attributes, named 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

The Year, Quarter, and Month attributes are used to construct a user-defined hierarchy, named Calendar, in the time dimension. The relationship between the levels and members of the Calendar dimension (a regular dimension) is shown in the following diagram.



Note:

Any hierarchy other than the default two-level attribute hierarchy is called a user-defined hierarchy. For more information about attribute hierarchies, see Attributes and Attribute Hierarchies.

Member Structures

With the exception of parent-child hierarchies, the positions of members within the hierarchy are controlled by the order of the attributes in the hierarchy's definition. Each attribute in the hierarchy definition constitutes a level in the hierarchy. The position of a member within a level is determined by the ordering of the attribute used to create the level. The member structures of user-defined hierarchies can take one of four basic forms, depending on how the members are related to each other.

Balanced 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. The Product Categories hierarchy of the Product dimension in the **Adventure Works DW** sample Analysis Services database is a good example of a balanced hierarchy. Each member in the Product Name level has a parent member in the Subcategory level, which in turn has a parent member in the Category level. Also, every branch in the hierarchy has a leaf member in the Product Name level.

Unbalanced Hierarchies

In an unbalanced hierarchy, branches of the hierarchy descend to different levels. Parent-child hierarchies are unbalanced hierarchies. For example, the Organization dimension in the **Adventure Works DW** sample Analysis Services database contains a member for each employee. 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.
It may be impossible for end users to distinguish between unbalanced and ragged hierarchies. However, you employ different techniques and properties in Analysis Services to support these two types of hierarchies. For more information, see Working with Ragged Hierarchies, and Working with Attributes in Parent-Child Hierarchies.

Ragged Hierarchies

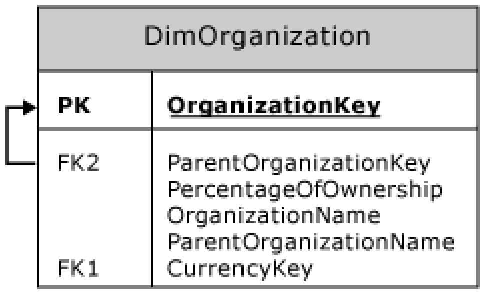
In a ragged hierarchy, the logical parent member of at least one member is not in the level immediately above the member. This can cause branches of the hierarchy to descend to different levels. For example, in a Geography dimension defined with the levels Continent, CountryRegion, and City, in that 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 Vatican City member is added to the CountryRegion level.
- Members are added to the City level and are associated with the Vatican City member in the CountryRegion level.
- A level, named Province, is added between the CountryRegion and City levels.

The Province level is populated with members associated with other members in the CountryRegion level, and members in the City level are associated with their corresponding members in the Province level. However, because the Vatican City member in the CountryRegion level has no associated members in the Province level, members must be associated from the City level directly to the Vatican City member in the CountryRegion level. Because of the changes, the hierarchy of the dimension is now ragged. The parent of the city Vatican City is the country/region Vatican City, which is not in the level immediately above the Vatican City member in the City level. For more information, see Working with Ragged Hierarchies.

Parent-Child Hierarchies

Parent-child hierarchies for dimensions are defined by using a special attribute, called a parent attribute, to determine how members relate to each other. A parent attribute describes a *self-referencing relationship*, or *self-join*, within a dimension main table. Parent-child hierarchies are constructed from a single parent attribute. Only one level is assigned to a parent-child hierarchy, because the levels present in the hierarchy are drawn from the parent-child relationships between members associated with the parent attribute. The dimension schema of a parent-child hierarchy depends on a self-referencing relationship present on the dimension main table. For example, the following diagram illustrates the **DimOrganization** dimension main table in the ***AdventureWorksDW*** Analysis Services sample database.



In this dimension table, the **ParentOrganizationKey** column has a foreign key relationship with the **OrganizationKey** primary key column. In other words, each record in this table can be related through a parent-child relationship with another record in the table. This kind of self-join is generally used to represent organization entity data, such as the management structure of employees in a department.
When you create a parent-child hierarchy, the columns represented by both attributes must have the same data type. Both attributes must also be in the same table. 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).
The depth of a parent-child hierarchy can vary among its hierarchical branches. In other words, a parent-child hierarchy is considered an unbalanced hierarchy.
Unlike user-defined hierarchies, in which the number of levels in the hierarchy determines the number of levels that can be seen by end users, a parent-child hierarchy is defined with the single level of an attribute hierarchy, and the values in this single level produce the multiple levels seen by users. The number of displayed levels depends on the contents of the dimension table columns that store the member keys and the parent keys. The number of levels can change when the data in the dimension tables change.

Write-Enabled Dimensions

Sunday, August 03, 2008 2:45 AM

Write-Enabled Dimensions

The data in a dimension is generally read-only. However, for certain scenarios, you may want to write-enable a dimension. In Microsoft SQL Server 2005 Analysis Services (SSAS), write-enabling a dimension enables business users to modify the contents of the dimension and see the immediate affect of changes on the hierarchies of the dimension. Any dimension that is based on a single table can be write-enabled. In a write-enabled dimension, business users and administrators can change, move, add, and delete attribute members within the dimension. These updates are referred to collectively as *dimension writeback*.

Analysis Services supports dimension writeback on all dimension attributes and any member of a dimension may be modified. For a write-enabled cube or partition, updates are stored in a writeback table separate from the cube's source tables. However, for a write-enabled dimension, updates 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 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 business users the ability to update members, whereas a write-enabled cube gives them the ability to update cell values. Although these two features are complementary, you do not have to use both features in combination. A dimension does not have to be included in a cube for dimension writeback to occur. A write-enabled dimension can also be included in a cube that is not write-enabled. You use different procedures to write-enable dimensions and cubes, and to maintain their security.

The following restrictions apply to dimension writeback:

- When you create a new member, you must include every attribute in a dimension. You cannot insert a member without specifying a value for the key attribute of the dimension. Therefore, creating members is subject to any constraints (such as non-null key values) that are defined on the dimension table.
- Dimension writeback is supported only for star schemas. In other words, a dimension must be based on a single dimension table directly related to a fact table. After you write-enable a dimension, Analysis Services validates this requirement when you deploy to an existing Analysis Services database or when you build an Analysis Services project.

Any existing member of a writeback dimension can be modified or deleted. When a member is deleted, the deletion cascades to all child members. For example, in a Customer dimension that contains CountryRegion, Province, City, and Customer attributes, deleting a country/region would delete all provinces, cities, and customers that belong to the deleted country/region. If a country/region has only one province, deleting that province would delete the country/region also.

Members of a writeback dimension can only be moved within the same level. For example, a city could be moved to the City level in a different country/region or province, but a city cannot be moved to the Province or CountryRegion level. In a parent-child hierarchy, all members are leaf members, and therefore a member may be moved to any level other than the **(All)** level.

If a member of a parent-child hierarchy is deleted, the member's children are moved to the member's parent. Update permissions on the relational table are required on the deleted member, but no permissions are required on the moved members. When an application moves a member in a parent-child hierarchy, the application can specify in the UPDATE operation whether descendents of the member are moved with the member or are moved to the member's parent. To recursively delete a member in a parent-child hierarchy, a user must have update permissions on the relational table for the member and all the member's descendants.

Note:

Updates to the parent attribute in a parent-child hierarchy must not include updates to any other properties or attributes.

All changes to a dimension cause the dimension structure to be modified. Each change to a dimension is considered a single transaction, requiring incremental processing to update the dimension structure. Write-enabled dimensions have the same processing requirements as any other dimension.

Note:

Dimension writeback is not supported by linked dimensions.

Security

The only business users who can update a write-enabled dimension are those in Analysis Services database roles that have been granted read/write permission to the dimension. For each role, you can control which members can and cannot be updated. For business users to update write-enabled dimensions, their client application must support this capability. For such users, a write-enabled dimension must be included in a cube that was processed since the dimension last changed.
Users and groups included in the Administrators role can update the attribute members of a write-enabled dimension, even if the dimension is not included in a cube.

Dimension Translations

Sunday, August 03, 2008 2:48 AM

Dimension Translations
In Microsoft SQL Server 2005 Analysis Services (SSAS), a dimension translation is a language -specific representation of the name of a dimension, the name of an Analysis Services object or one of its members, such as a caption, member, or hierarchy level. SQL Server 2005 Ana lysis Services also supports translations of cube objects.
Translations provide server support for client applications that can support multiple languages. Frequently, users from diffe rent countries view a cube and its dimensions. It is useful to be able to translate various elements of a cube and its dimensions into a different langu age so that these users can view and understand the cube. For example, a business user in France can access a cube from a workstation with a French local e setting, and see the object property values in French. However, a business user in Germany who accesses the same cube from a workstation with a Ge rman locale setting sees the same object property values in German.
The collation and language information for the client computer is stored in the form of a locale identifier (LCID). Upon conn ection, the client passes the LCID to the instance of Analysis Services. The instance uses the LCID to determine which set of translations to use when prov iding metadata for Analysis Services objects. If an Analysis Services object does not contain the specified translation, the default language is used to return the content back to the client.

>

Mining Structures

Sunday, August 03, 2008 2:49 AM

Mining Structures (Analysis Services)

There are several objects involved in data mining in Microsoft SQL Server 2005 Analysis Services (SSAS). Following are the two primary objects that are used:

- Data mining structure
- Data mining model

The other objects involved in data mining are mining structure columns and mining model columns.

Data Mining Structure

The mining structure is a data structure that defines the data domain from which mining models are built. A single mining structure can contain multiple mining models that share the same domain. The building blocks of the mining structure are the mining structure columns, which describe the data that the data source contains. These columns contain information such as data type, content type, and how the data is distributed.

A mining structure can also contain nested tables. A nested table represents a one-to-many relationship between the entity of a case and its related attributes. For example, if the information that describes the customer resides in one table, and the customer's purchases reside in another table, you can use nested tables to combine the information into a single case. The customer identifier is the entity, and the purchases are the related attributes.

The mining structure does not contain information about how columns are used for a specific mining model, or about the type of algorithm that is used to build a model; this information is defined in the mining model itself.

Data Mining Model

A data mining model applies a mining model algorithm to the data that is represented by a mining structure. Like the mining structure, the mining model contains columns. A mining model is contained within the mining structure, and inherits all the values of the properties that are defined by the mining structure. The model can use all the columns that the mining structure contains or a subset of the columns.

In addition to the parameters that are defined on the mining structure, the mining model contains two properties: Algorithm and Usage. The a*lgorithm* parameter is defined on the mining model, and the u*sage* parameter is defined on the mining model column. These parameters are described in the following table.

algorithm	A model property that defines the algorithm that is used to create the model.
usage	A model column property that defines how a column is used by the model. You can define columns to be input columns, key columns, or predictable columns.

A data mining model is just an empty object until it is processed. When you process a model, the data that is defined by the structure is passed through the algorithm. The algorithm identifies rules and patterns within the data, and then uses these rules and patterns to populate the model. After you have processed a model, you can explore it by using the custom viewers that are provided in Business Intelligence Development Studio and SQL Server Management Studio, or by querying the model to perform predictions.

You can create multiple models that are based on the same structure. All models built from the same structure must be from the same data source. However, the models can differ as to which columns of the structure are used, how the columns are used, the type of algorithm that is used to create each model, and the parameter settings for each algorithm. For example, you can build separate decision tree and clustering models, each containing different columns from the structure and being used to accomplish different business tasks.

Roles

Sunday, August 03, 2008 2:53 AM

Roles (Analysis Services)
Roles are used in Microsoft SQL Server 2005 Analysis Services (SSAS) to manage security for Analysis Services objects and data. In basic terms, a role associates the security identifiers (SIDs) of Microsoft Windows users and groups that have specific access rights and permissions defined for objects managed by an instance of Analysis Services. Two types of roles are provided in Analysis Services:

- The server role, a fixed role that provides administrator access to an instance of Analysis Services.
- Database roles, roles defined by administrators to control access to objects and data for non-administrator users.

Server Role
The Analysis Services server role defines administrative access of Windows users and groups to an instance of Analysis Services. Members of this role have access to all Analysis Services databases and objects on an instance of Analysis Services, and can perform the following tasks:

- Perform server-level administrative functions using SQL Server Management Studio or Business Intelligence Development Studio, including creating databases and setting server-level properties.
- Perform administrative functions programmatically with Analysis Management Objects (AMO).
- Maintain Analysis Services database roles.
- Start traces (other than for processing events, which can be performed by a database role with Process access).

Every instance of Analysis Services has a server role that defines which users can administer that instance. The name and ID of this role is Administrators, and unlike database roles, the server role cannot be deleted, nor can permissions be added or removed. In other words, a user either is or is not an administrator for an instance of Analysis Services, depending on whether he or she is included in the server role for that instance of Analysis Services.

Database Roles
An Analysis Services database role defines user access to objects and data in an Analysis Services database. A database role is created as a separate object in an Analysis Services database, and applies only to the database in which that role is created. Windows users and groups are included in the role by an administrator, who also defines permissions within the role.
The permissions of a role may allow members to access and administer the database, in addition to the objects and data within the database. Each permission has one or more access rights associated with it, which in turn give the permission finer control over access to a particular object in the database.

XMLA

Sunday, August 03, 2008 3:09 AM

XML for Analysis (XMLA)

XML for Analysis (XMLA) is a Simple Object Access Protocol (SOAP)-based XML protocol, designed specifically for universal data access to any standard multidimensional data source residing on the Web.

XMLA also eliminates the need to deploy a client component that exposes Component Object Model (COM) or Microsoft .NET Framework interfaces. XMLA is optimized for the Internet, when round trips to the server are expensive in terms of time and resources, and when stateful connections to a data source can limit user connections on the server.
XMLA is the native protocol for Microsoft SQL Server 2005 Analysis Services (SSAS), used for all interaction between a client application and an instance of Analysis Services. Analysis Services fully supports XML for Analysis 1.1, and also provides extensions to support metadata management, session management, and locking capabilities. Both Analysis Management Objects (AMO) and ADOMD.NET use the XMLA protocol when communicating with an instance of Analysis Services.

In This Section

Topic	**Description**
XML for Analysis Overview (XMLA)	Provides a brief discussion of the XMLA protocol
Using XML for Analysis in Analysis Services (XMLA)	Describes how to use XMLA to manipulate data and metadata.
XML for Analysis Reference (XMLA)	Details the XMLA protocol as implemented in Analysis Services.

⊟
Se