|  |
| --- |
|  |
| SQL SERVER ANALYSIS SERVICES |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |

SSAS OVERVIEW

[Overview of OLAP 5](#_Toc343269829)

[Understanding SSAS & Components 7](#_Toc343269830)

[OLAP 7](#_Toc343269831)

[Data Mining 7](#_Toc343269832)

[Cube 8](#_Toc343269833)

[Dimension table 8](#_Toc343269834)

[Dimension 8](#_Toc343269835)

[Hierarchy 9](#_Toc343269836)

[Level 9](#_Toc343269837)

[Fact table 9](#_Toc343269838)

[Measure 9](#_Toc343269839)

[Schema 9](#_Toc343269840)

[History of SSAS 10](#_Toc343269841)

[Defining SSAS Data Warehouse 11](#_Toc343269842)

[Data Warehouse Start Schema in SSAS 11](#_Toc343269843)

[SSAS Star Schema 11](#_Toc343269844)

[Data Warehouse Snowflake Schema in SSAS 12](#_Toc343269845)

[Architecture of SSAS 13](#_Toc343269846)

[Analysis Services Architecture 13](#_Toc343269847)

[Server Architecture 13](#_Toc343269848)

[Client Architecture 14](#_Toc343269849)

[Analysis Services Objects 15](#_Toc343269850)

[1) Data Sources 15](#_Toc343269851)

[2) Data Source Views 15](#_Toc343269852)

[3) Cubes 16](#_Toc343269853)

[4) Dimensions 17](#_Toc343269854)

[5) Mining structures 18](#_Toc343269855)

[6) Roles 18](#_Toc343269856)

[7) Assemblies 18](#_Toc343269857)

[Creating Cube in SSAS 2008 21](#_Toc343269858)

[1. Creating Analysis Services Project. 22](#_Toc343269859)

[2. Creating Data Source 22](#_Toc343269860)

[3. Creating Data Source View 23](#_Toc343269861)

[4. Creating Cube and Dimensions 24](#_Toc343269862)

[5. Creating Dimension Hierarchies 26](#_Toc343269863)

[6. Deploying Cube Database from BIDS 27](#_Toc343269864)

[Structure of Cube 29](#_Toc343269865)

[Cube Structure 29](#_Toc343269866)

[Dimension Usage 29](#_Toc343269867)

[Calculations 29](#_Toc343269868)

[KPIs 29](#_Toc343269869)

[Actions 29](#_Toc343269870)

[Partitions 30](#_Toc343269871)

[Perspectives 30](#_Toc343269872)

[Translations 30](#_Toc343269873)

[Browser 30](#_Toc343269874)

[**Accessing Cubes from Excel** 31](#_Toc343269875)

[Introduction 31](#_Toc343269876)

[Review Cube and Dimension properties in Cube Designer 38](#_Toc343269877)

[1. Measure Pane 38](#_Toc343269878)

[2. Dimension Pane 38](#_Toc343269879)

[Introduction to MDX 44](#_Toc343269880)

[Sample cube 45](#_Toc343269881)

[Understanding MDX query 45](#_Toc343269882)

[MDX & SQL QUERY 45](#_Toc343269883)

[Cells, Tuples, and Sets 47](#_Toc343269884)

[Tuple 47](#_Toc343269885)

[Axis and Slicer Dimensions 48](#_Toc343269886)

[MDX QUERY SAMPLES 48](#_Toc343269887)

[FUNCTIONS 52](#_Toc343269888)

[1. COUNT FUNCTION 52](#_Toc343269889)

[2. IIF functions 53](#_Toc343269890)

[CALCULATED MEMBERS 54](#_Toc343269891)

[CALCULATED MEMBERS 56](#_Toc343269892)

[Calculated Members 57](#_Toc343269893)

[CREATING A CALCULATED MEMBER: 57](#_Toc343269894)

[Overview of KPI 61](#_Toc343269895)

[Creating a Key Performance Indicator 62](#_Toc343269896)

[Adding the Value Expression 63](#_Toc343269897)

[Adding the Goal Expression 64](#_Toc343269898)

[Adding the Status Expression 64](#_Toc343269899)

[Adding the Trend Expression 66](#_Toc343269900)

[Completing the Key Performance Indicator 68](#_Toc343269901)

[SSAS Actions 71](#_Toc343269902)

[Creating a DrillDown Action 71](#_Toc343269903)

[Step 1 71](#_Toc343269904)

[Step 2 71](#_Toc343269905)

[Step 3 72](#_Toc343269906)

[Step 4 73](#_Toc343269907)

[Step 5 73](#_Toc343269908)

[SSAS Perspectives 75](#_Toc343269909)

[Perspectives vs. Views 75](#_Toc343269910)

[Working with Perspectives 75](#_Toc343269911)

[Creating A SSAS 2008-Perspectives 76](#_Toc343269912)

[Overview of Translation 77](#_Toc343269913)

[Creating a Translation in SSAS 77](#_Toc343269914)

[Partition 84](#_Toc343269915)

[Creating a Partition in SSAS 85](#_Toc343269916)

[WRITEBACKS IN CUBES 88](#_Toc343269917)

[PROACTIVE CACHING 94](#_Toc343269918)

[Real-time HOLAP 94](#_Toc343269919)

[**Real-time ROLAP** 94](#_Toc343269920)

[Configuring proactive caching 94](#_Toc343269921)

[Aggregations 102](#_Toc343269922)

# SSAS IS AN ONLINE ANAYTICAL PROCESSING

# AND DATA MINING TOOL OF SQL SERVER

# Overview of OLAP

OLAP stands for On Line Analytical Processing, a series of protocols used mainly for business reporting.

Using OLAP, businesses can analyze data in all manner of different ways, including [budgeting](http://www.wisegeek.com/what-is-budgeting.htm), planning, simulation, data [warehouse](http://www.wisegeek.com/what-is-a-warehouse.htm) reporting, and trend analysis.

A main component of OLAP is its ability to make multidimensional calculations, allowing a wide and lightning-fast array of possibilities.

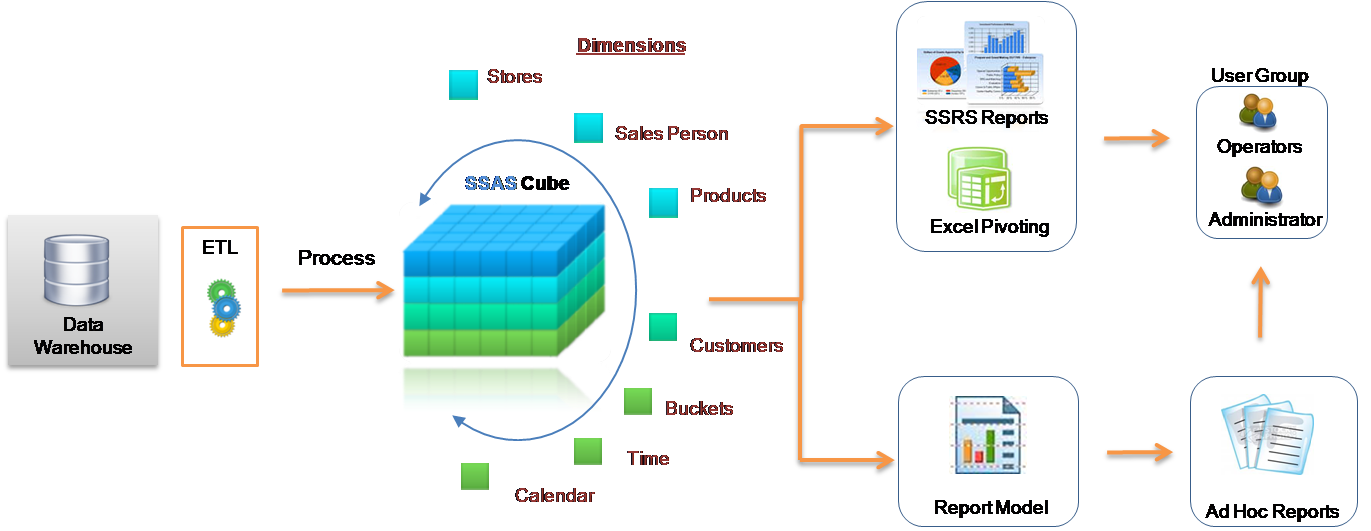
One of the effects of business growth is a mountain of data that has to be manufactured, stored, tracked, and interpreted. A growing business needs sophisticated methods of [data processing](http://www.wisegeek.com/what-is-data-processing.htm). One of the ways that businesses do this is to use OLAP.

In addition, the bigger the business, the bigger its business reporting needs. Multidimensional calculations enable a large business to complete in seconds what it otherwise would have waited a handful of minutes to receive.

Benefits of OLAP

* An OLAP cube is a technology that stores data in an optimized way to provide a quick response to various types of complex queries by using dimensions and measures.  
  Most cubes store pre-aggregates of the measures with its special storage structure to provide quick response to queries.
* SSRS Reports and Excel Power Pivot is used as front end for Reporting and data analysis with SSAS (SQL Server Analysis Services) OLAP Cube.
* SSAS (SQL Server Analysis Services) is Microsoft BI Tool for creating Online Analytical Processing and data mining functionality.
* BIDS (Business Intelligence Development Studio) provides environment for developing your OLAP Cube and Deploy on SQL Server.
* BIDS (Business Intelligence Development Studio) comes with Microsoft SQL Server 2005, 2008 (e.g. Developer, Enterprise Edition) .
* We have to choose OLAP Cube when performance is a key factor, the key decision makers of the company can ask for statistics from the data anytime from your huge database.
* We can perform various types of analysis on data stored in Cube, it is also possible to create data mining structure on this data which can be helpful in forecasting, prediction.

**Basic Architecture**

* In our case, data warehouse is used as a source of data to Cube in BIDS. Once Cube gets ready with data, users can run queries on Cube created in SSAS. SSRS Reports and Excel Pivoting/Power Pivot can use OLAP Cube as source of data instead of OLTP database to get performance for resolving Complex Queries.
* SSRS Reports, Excel Power Pivot can be used for visualization/analysis of data from cube.
* 

One main benefit of OLAP is consistency of calculations. No matter how fast data is processed through OLAP software or servers, the reporting that results is presented in a consistent presentation, so executives always know what to look for where. This is especially helpful when comparing information from previous reports to information contained in new ones and projected future ones. "What if" scenarios are some of the most popular uses of OLAP software and are made eminently more possible by multidimensional processing.

**Another benefit of multidimensional data presentation is that it allows a** [**manager**](http://www.wisegeek.com/what-is-a-manager.htm) **to pull down data from an OLAP database in broad or specific terms.** In other words, reporting can be as simple as comparing a few lines of data in one column of a spreadsheet or as complex as viewing all aspects of a mountain of data.

Also, multidimensional presentation can create an understanding of relationships not previously realized. All of this, of course, can be done in the blink of an eye.

Producers of OLAP software are familiar, including Oracle, IBM, and Hyperion Solutions. Oracle, which has a reputation for being different, refers to OLAP software as Business Intelligence. IBM and Hyperion Solutions, wishing to remain consistent with industry standards, call their software OLAP.

An Online Analytical Processing (OLAP) cube is a powerful tool to analyze large amounts of data quickly. There are a number of products available that can create an OLAP cube. Factors that should be considered in choosing an OLAP technology and evaluate two major vendors’ products – Microsoft SQL Server Analysis Services 2008 (SSAS) and IBM Cognos Power Cube.

Three factors that are important in considering OLAP technologies are:  
1. Your existing database and reporting technologies.  
2. The amount of data that will be brought into the cube.  
3. The skill set of workers that will design and create cubes.

**Microsoft SQL Server**

If your data warehouse platform is based on Microsoft SQL Server, especially if you are also using SQL Server Integration Services (SSIS), then SSAS is a natural extension of the architecture. SSIS can used to load and process the SSAS cube. This Microsoft database environment can also be used as source data for an IBM Cognos cube, but the Cognos cube cannot be used with Microsoft reporting tools such as Reporting Services. IBM Cognos does have a software product that allows Powercube browsing with Excel; Microsoft SSAS cubes can be browsed natively with Excel pivot tables.

**SharePoint**  
If your portal and collaboration solution is Microsoft SharePoint there, are exciting features in SharePoint 2010 that combines Performance Point and new PowerPivot software. These will leverage SSAS cubes and will not support IBM Cognos Power cubes.

**IBM Cognos environment**  
If your database platform is not Microsoft and your reporting environment is purely IBM Cognos Business Intelligence, then using Powercubes is a natural extension. Analysis Services cubes may also be used in this environment if other considerations make this desirable, but all things being equal, Cognos is a better fit.

**Size of data**  
The size of the source data and resulting cube is of paramount importance in your choice.

**CognosPowercubes have an inherent limit of 2 GB, although there are workaround techniques.**

Microsoft SSAS cubes are commonly 300-400 GB in size with source data measured in terabytes.

Multi-terabyte SSAS cubes are in use today. SSAS also gives the ability to use relational tables for aggregation (known as ROLAP) or a hybrid (known as HOLAP). This allows for even more scalability. For large amounts of data, Microsoft is a clear winner.

**Learning Curve**  
Microsoft SSAS requires a more technical skill set for developers than IBM CognosPowercubes. Microsoft cubes will require a working knowledge of the multi-dimensional language, MDX. For developers that know SQL it will look similar, but is a paradigm shift analogous to moving from procedural languages to object oriented languages. There will be a learning curve. The return for that investment is much more flexibility, programmability and extensibility.  
IBM Cognos Transformer, the software used to design a Powercube, was designed with the developer or power user in mind. IBM envisions strong financial analysts creating their own cubes. The result is simpler and easier to use but lacks the rich capabilities present in SSAS. Organizations with a medium amount of data and limited technical resources can build solutions quicker using IBM Cognos cubes.

# Understanding **SSAS**& Components

SQL Server Analysis Services (SSAS) is the technology from the Microsoft Business Intelligence stack, to develop Online Analytical Processing (OLAP) solutions.

In simple terms, you can use SSAS to create cubes using data from data marts / data warehouse for deeper and faster data analysis.

Microsoft SQL Server Analysis Services (SSAS) delivers both online analytical processing (OLAP) and data mining functionality for business intelligence applications.

## OLAP

Online analytical processing (OLAP) allows the user to access aggregated and organized data from business data sources, such as data warehouses, in a multidimensional structure called a cube. Microsoft provides tools and features for OLAP that user can use to design, deploy, and maintain cubes and other supporting objects.

## Data Mining

Data mining gives the user access to the information that is needed to make intelligent decisions about difficult business problems. Microsoft provides tools for data mining with which user can identify rules and patterns in the data, so that the user can determine why things happen and predict what will happen in the future.

Cubes are multi-dimensional data sources which have dimensions and facts (also known as measures) as its basic constituents. From a relational perspective dimensions can be thought of as master tables and facts can be thought of as measureable details. These details are generally stored in a pre-aggregated proprietary format and users can analyze huge amounts of data and slice this data by dimensions very easily. Multi-dimensional expression (MDX) is the query language used to query a cube, similar to the way T-SQL is used to query a table in SQL Server.

Simple examples of dimensions can be product / geography / time / customer, and similar simple examples of facts can be orders / sales. A typical analysis could be to analyze sales in Asia-pacific geography during the past 5 years. You can think of this data as a pivot table where geography is the column-axis and years is the row axis, and sales can be seen as the values. Geography can also have its own hierarchy like Country->City->State.  Time can also have its own hierarchy like Year->Semester->Quarter. Sales could then be analyzed using any of these hierarchies for effective data analysis.

A typical higher level cube development process using SSAS involves the following steps:  
  
1) Reading data from a dimensional model  
2) Configuring a schema in BIDS (Business Intelligence Development Studio)  
3) Creating dimensions, measures and cubes from this schema  
4) Fine tuning the cube as per the requirements  
5) Deploying the cube

The basic concepts of OLAP include:

* Cube
* Dimension table
* Dimension
* Hierarchy
* Level
* Fact table
* Measure
* Schema

## Cube

The basic unit of storage and analysis in Analysis Services is the *cube*. **A cube is a collection of data that's been aggregated to allow queries to return data quickly**. For example, a cube of order data might be aggregated by time period and by title, making the cube fast when you ask questions concerning orders by week or orders by title.

**Cubes are ordered into *dimensions* and *measures*.**

The data for a cube comes from a set of staging tables, sometimes called a star-schema database. Dimensions in the cube come from *dimension tables* in the staging database, while measures come from *fact tables* in the staging database.

## Dimension table

**A *dimension table* lives in the staging database and contains data that you'd like to use to group the values you are summarizing.** Dimension tables contain a primary key and any other attributes that describe the entities stored in the table. Examples would be a Customers table that contains city, state and postal code information to be able to analyze sales geographically, or a Products table that contains categories and product lines to break down sales figures.

## Dimension

**Each cube has one or more *dimensions*, each based on one or more dimension tables.** A dimension represents a category for analyzing business data: time or category in the examples above. Typically, a dimension has a natural hierarchy so that lower results can be "rolled up" into higher results. For example, in a geographical level you might have city totals aggregated into state totals, or state totals into country totals.

## Hierarchy

**A *hierarchy* can be best visualized as a node tree.** A company's organizational chart is an example of a hierarchy. Each dimension can contain multiple hierarchies; some of them are *natural* hierarchies (the parent-child relationship between attribute values occur naturally in the data), others are *navigational* hierarchies (the parent-child relationship is established by developers.)

## Level

**Each layer in a hierarchy is called a *level*.** For example, you can speak of a week level or a month level in a fiscal time hierarchy, and a city level or a country level in a geography hierarchy.

## Fact table

**A *fact table* lives in the staging database and contains the basic information that you wish to summarize.** This might be order detail information, payroll records, drug effectiveness information, or anything else that's amenable to summing and averaging. Any table that you've used with a Sum or Avg function in a totals query is a good bet to be a fact table. The fact tables contain fields for the individual facts as well as foreign key fields relating the facts to the dimension tables.

## Measure

Every cube will contain one or more *measures*, each based on a column in a fact table that you';d like to analyze. In the cube of book order information, for example, the measures would be things such as unit sales and profit.

## Schema

Fact tables and dimension tables are related, which is hardly surprising, given that you use the dimension tables to group information from the fact table. The relations within a cube form a *schema*. There are two basic OLAP schemas: star and snowflake. In a *star schema*, every dimension table is related directly to the fact table. In a *snowflake schema*, some dimension tables are related indirectly to the fact table. For example, if your cube includes OrderDetails as a fact table, with Customers and Orders as dimension tables, and Customers is related to Orders, which in turn is related to OrderDetails, then you're dealing with a snowflake schema.

# History of SSAS

SSAS takes his root from OLAP technology developed by Panorama Software which Microsoft acquired in 1996. Since that time Microsoft diligently worked on SSAS and shipped it with SQL Server 7 and SQL Server 2000. However, **SSAS really took off in terms of adaptation in SQL Server 2005**. It was for the first time when SSAS was accessible from within SQL Server Management Studio and integrated with Business Intelligence Development Studio.

Other major enhancements introduced in SQL server 2005 and SSAS such as

* Easier way of configuring SSAS.
* Introduction of Data Source View (DSV) - is designed as a mapping tool between relational database and Cube solution, it also allowed unification of data from multiple sources.
* Unified Dimensional Model (UDM) - is designed as a virtual model for a database warehouse.
* Data mining add-in for Excel – is designed to allow Excel to play with the data in a more powerful way.
* Key Performance Indicator (KPI) – is designed to allow definition of a current data point, a target data point and trending indicators.
* Business Intelligence Development Studio (BIDS) – is designed to peer into cubes and examine data.

Microsoft did a good job with SQL Server 2008 by polishing existing setoff tools related to SSAS and taking into account user’s suggestions and asks. However, there was little structurally new introduced with SQL Server 2008. It is considered to be more of an evolutionary release for SSAS rather than revolutionary.

## Defining SSAS Data Warehouse

SSAS Cubes rely on relational data structure. Essentially, the SSAS Cubes is a collection of tables and relationships in DBMS. However, nomenclature is different in the OLAP world. We call tables not just entity tables but fact tables and dimension tables.

* Fact Table is a relational database table that houses a measure.
* Dimensions Table is a relational database table that houses a lookup data.
* Combination of Fact and Dimensions can be stored in one table and this table is also called Fact Table.

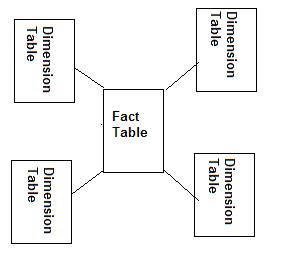
There are two different data warehouse schemas exist in the OLAP world. First schema is called star schema and it resembles star like structure. Second schema is called snowflake schema and it resembles a snowflake.

## Data Warehouse Start Schema in SSAS

## SSAS Star Schema

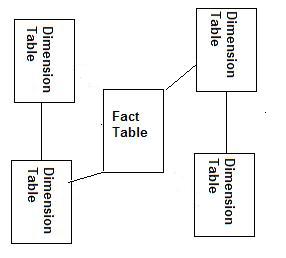
**SSAS Star Schema is designed with one table in the center or SSAS Fact Table and lookup tables or SSAS Dimension Tables linked to this central table.**

SSAS Star schemas are easier to visualize and it is faster to retrieve data from the SSAS start schema. However, there is a need to flatten data or de-normalize and it takes more space on the disk. There are products that are configured to use only start schemas and not the others. So it may be a natural choice to use just this type.

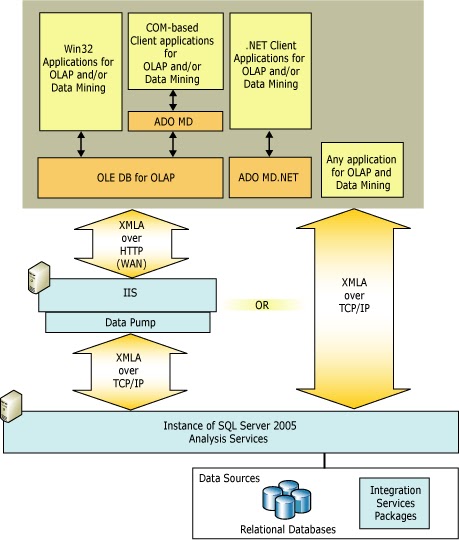


## Data Warehouse Snowflake Schema in SSAS

**SSAS Snowflake Schema is any, other than start, schema design.** However, it is usually normalized. Data hierarchies are defined with the help of multiple related tables with the help of primary and foreign key relationships. SSAS Snowflake schemas contain multiple fact and dimensional tables. The SSAS Snowflake schemas have few advantages. They require less storage space due to highly normalized structure of its scheme. However, they are slower for data processing due to multiple joins which take most of the computational power. They are also harder to update.



# Architecture of SSAS



## 

## Analysis Services Architecture

SSAS use server and client components to supply OLAP and data mining.

### Server Architecture

The server component of SSAS is the **msmdsrv.exe application, which runs as a Windows service**. SSAS **supports multiple instances** on the same computer, with each instance of Analysis Services implemented as a separate instance of the Windows service. The msmdsrv.exe application consists of security components, an XML for Analysis (XMLA) listener component, a query processor component and numerous other internal components that perform the following functions:

* Parsing statements received from clients
* Managing metadata
* Handling transactions
* Processing calculations
* Storing dimension and cell data
* Creating aggregations
* Scheduling queries
* Caching objects
* Managing server resources

The XMLA listener component handles all XMLA communications between Analysis Services and its clients. The Analysis Services Port configuration setting in the msmdsrv.ini file can be used to specify a port on which an Analysis Services instance listens. **A value of 0 in this file indicates that Analysis Services listen on the default port**. The default instance of Analysis Services listens on TCP/IP port 2383, but named instances of Analysis Services do not use a default port. Each named instance listens either on the port that an Administrator specifies, or on the port that is dynamically assigned at startup. This variability among ports means that clients do not automatically know which port a particular named instance of Analysis Services is using, and therefore do not automatically know where to send their requests. To make it easy for clients to send requests to named instances of Analysis Services, SQL Server has a service called SQL Server Browser. SQL Server Browser keeps track of the ports on which each named instance listens. Client connection requests for a named instance that do not specify a port number are directed to port 2382, the port on which SQL Server Browser listens. SQL Server Browser then redirects the request to the port that the named instance uses.

### Client Architecture

Client components 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.

Several different providers are provided with Analysis Services to support different programming languages. A provider communicates with an Analysis Services server by sending and receiving XML for Analysis in SOAP packets over TCP/IP or over HTTP through Internet Information Services (IIS). An HTTP connection uses a COM object instantiated by IIS, called a data pump, which acts as a conduit for Analysis Services data.

Win32 client applications can connect to an Analysis Services server using OLE DB for OLAP interfaces or the ActiveX Data Objects (ADO) object model for Component Object Model (COM) automation languages, such as Visual Basic. Applications coded with .NET languages can connect to an Analysis Services server using ADO MD.NET.

Query commands can be issued using the following languages:

1) SQL

2) Multidimensional Expressions (MDX), an industry standard query language for analysis

3) Data Mining Extensions (DMX), an industry standard query language for data mining.

4) Analysis Services Scripting Language (ASSL) can also be used to manage Analysis Services database objects.

**Unified Dimension Model**

Analysis Services combines the best aspects of traditional OLAP-based analysis and relational-based reporting by enabling developers to define a single data model, called a Unified Dimensional Model (UDM) over one or more physical data sources. All end user queries from OLAP, reporting, and custom BI applications access the data in the underlying data sources through the UDM, which provides a single business view of this relational data. The UDM is a central place that serves as the single version of truth for all reports, spreadsheets, OLAP browsers, KPIs, and analytical applications.

# Analysis Services Objects

A 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.

## 1) Data Sources

A data source in Microsoft represents a connection to any data source(Oracle,DB2,Teradata) 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.

## 2) Data Source Views

A data source view contains the logical model of the schema used by Analysis Services database objects—namely cubes, dimensions, and mining structures. A data source view is the metadata definition, stored in an XML format, of these schema elements used by the Unified Dimensional Model (UDM) and by the mining structures.

## 3) Cubes

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) Calculation

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 Analysis Services.

B) KPI (Key performance indicator)

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.

C) Actions

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.

D) Partitions

Partitions are used by Analysis Services to manage and store data and aggregations for a measure group in a cube. Partitions are a powerful and flexible means of managing cubes, especially large cubes.

E) Perspective

**Cubes can be very complex objects for users to explore in 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.

One 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

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.

**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.

F) Cube translations

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.

## 4) Dimensions

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 are groups of attributes based on columns from tables or views in a data source view.

## 5) Mining structures

There are several objects involved in data mining in SSAS. Following are the two primary objects that are used:

* Data mining structure
* Data mining model

A) 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.

B) 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.

## 6) Roles

Roles are used 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.

## 7) Assemblies

|  |
| --- |
| Analysis Services lets the user add assemblies to an Analysis Services instance or database. Assemblies lets the user create external, user-defined functions using any common language runtime (CLR) language, such as Visual Basic .NET or Visual C#. |

# Creating Cube in SSAS 2012

Nowadays, analytical solutions are becoming mission critical for many organizations. Microsoft SQL Server 2008 Analysis Services (SSAS) is designed to provide exceptional performance and scalable support with millions of records and thousands of users from different locations.  
  
Why to Build a Cube?  
There are many advantages of cube over relational data mart.

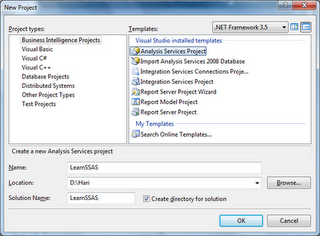
* While querying a data mart, you can get most of the results but not everything you need for business analysis and decision making. Cube can help you to get answers of all "What-If" scenarios.
* Building a cube helps to house your data to centralize the business rules for calculations that you can't easily store in a relational data mart.
* The structure of the cube makes it much easier to write queries to compare data year over year (YOY), or to create cumulative values such as year-to-date (YTD) sales.
* Scalable Infrastructure - Analysis Services can scale to support databases of many terabytes in size with many thousands of users.
* Superior Performance - Analysis Services cubes are multidimensional structures that enable fast access to high volumes of pre-aggregated data, empowering end users to gain insight into relevant business data at the speed of thought.
* You gain the ability to manage aggregated data in the cube. To improve query performance in a relational data mart, we often create summary tables to prepare data for queries that don't require transaction-level detail. SSAS creates the logical equivalent of summary tables (called aggregations) and keeps them up-to-date.

Creating a Cube consists of the following steps :

1. Creating Analysis Services Project
2. Creating Data Source
3. Creating Data Source View
4. Creating Cube and Dimensions
5. Creating Dimension Hierarchies
6. Deploying Cube Database from BIDS

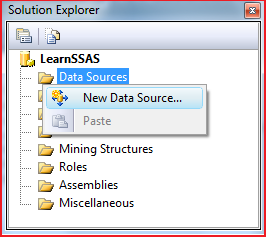
## Creating Analysis Services Project.

First step is to create a project in Business Intelligence Development Studio (BIDS). Launch BIDS from Start -->All Programs --> Microsoft SQL Server 2008 -->SQL Server Business Intelligence Development Studio and then click File -->New -->Project. In the *New Project* dialog box, select Analysis Services Project. In the Name text box, type **LearnSSAS** and, if you like, change the location for your project. I'll store this project at location *D:\Hari*. Click OK to create the project.

[](http://2.bp.blogspot.com/_2t15yG-FxQA/TEoKD8YyHHI/AAAAAAAAAy4/WJQStjUqJPA/s1600/1.png)

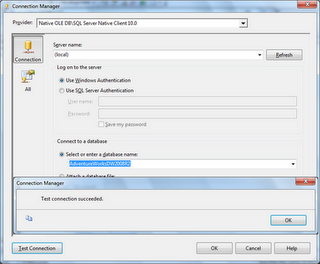
## Creating Data Source

Now add a data source to define the connection string for data mart **AdventureWorksDW2008R2**. In Solution Explorer, right-click the Data Sources folder and click New Data Source.

[](http://4.bp.blogspot.com/_2t15yG-FxQA/TEoUW2qcicI/AAAAAAAAAzA/IFRnDssrjvs/s1600/1.png)

In the Data Source Wizard, click Next on the Welcome to the Data Source Wizard page if it hasn't been disabled. On the Select how to define the connection page, click New to set up a new connection. In the Connection Manager, the default provider is the Native OLE DB\SQL Server Native Client 10.0, which is correct for our project.

To define the connection, type the name of your server in the Server Name text box. Alternatively you can select it from the drop-down list, then select AdventureWorksDW2008R2 in the database drop-down list and click Test Connection button to check the connection. Finally click on OK as shown below:

[](http://2.bp.blogspot.com/_2t15yG-FxQA/TEoXCsYjBJI/AAAAAAAAAzI/6jbILVZeEPw/s1600/1.png)

When you're back in the Data Source Wizard, click Next. On the Impersonation Information page, select **Use the service account option** so that service account will be used to read data from the source when loading data into your SSAS database and service account  must have read permissions to do so. Click Next and then Finish to complete the wizard.

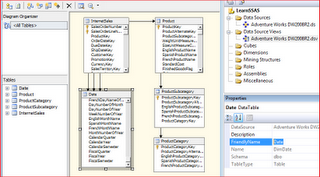
## Creating Data Source View

Now next step is to create a data source view (DSV) from the data source to define dimensions and cubes. You can make changes to the DSV without modifying the actual data source, which is very useful if you have only read permissions to the data mart. In Solution Explorer, right-click on the Data Source Views folder and then click New Data Source Views...You can see *Data Source View Wizard*. Click Next on the Welcome page. On the Select a Data Source page, select the data source just added to the project (Adventure Works DW2008R2.ds) and click Next. Now add required objects to the DSV by double-clicking each table or view on Select Tables and Views page. I want to add the following tables to the DSV to make it easy to understand for beginners:

DimDate, DimProduct, DimProductCategory, DimProductSubcategory, and FactInternetSales. You can always add more tables later if you want to explore advance BI questions. Now click Next in the *Data Source View Wizard* once you are finished adding required tables followed by click on Finish. You can give a name to your DSV before clicking Finish button.

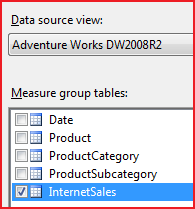
I would recommend you to change the name of objects by selecting each one in the DSV designer and remove the Dim and Fact prefixes from the FriendlyName property because when you create dimensions and cubes, only FriendlyName property will be assigned to the objects.

The DSV is shown below:

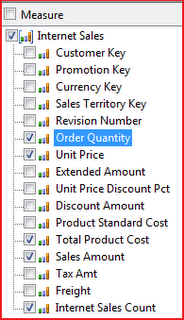
[](http://2.bp.blogspot.com/_2t15yG-FxQA/TEoiODj97EI/AAAAAAAAAzg/bI9Nf3W2oXk/s1600/1.png)

## Creating Cube and Dimensions

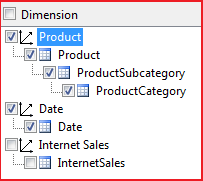
Next step is to create a Cube and Dimensions from the data source view. In *Solution Explorer*, right-click on the Cubes folder and then click New Cube...You can see Cube Wizard. Click on Next in Welcome to the Cube Wizardpage.On the Select Creation Method page, keep the default option Use existing tables and click Next button. On the Select Measure Group Tables page, choose **InternetSales** table and click Next.

[](http://3.bp.blogspot.com/_2t15yG-FxQA/TEonip9AoiI/AAAAAAAAAzo/BSeKvsEwYAE/s1600/1.png)

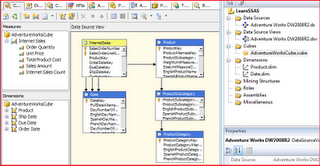
Now the wizard displays all the measures available in the selected measure group tables. Measures are basically numeric values e.g. OrderQuantity, Unit Price, Sales Amount, Tax Amount etc. Select only the following measures from Internet Sales Group: Order Quantity, Unit Price, Total Product Cost, Sales Amount, and Internet Sales Count

[](http://4.bp.blogspot.com/_2t15yG-FxQA/TEopZ-HwvVI/AAAAAAAAAzw/N_BsYpuGEvI/s1600/1.png)

Now click on Next button to open Select New Dimensions page and select Date and Product dimensions. Click Next to proceed.

[](http://4.bp.blogspot.com/_2t15yG-FxQA/TEoq5jd52wI/AAAAAAAAAz4/UN9b1sZkn04/s1600/1.png)

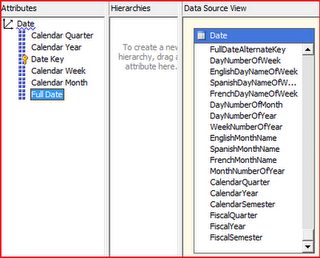
In the Completing the Wizard page, enter the cube name as **AdventureWorksCube** and click Finish button to complete the wizard. Cube layout is shown below:

[](http://2.bp.blogspot.com/_2t15yG-FxQA/TEoteTAgYpI/AAAAAAAAA0A/DKbxEmHfkiQ/s1600/1.png)

Now click on each dimensions and add required attributes from the Data Source View.

**Date Dimension:**

Drag and drop FullDateAlternateKey, CalendarYear, CalendarQuarter, EnglishMonthName, and EnglishDayNameOfWeek. Rename FullDateAlternateKey with Full Date, EnglishMonthName with Calendar Month, and EnglishDayNameOfWeek with Calendar Week as shown below:

[](http://4.bp.blogspot.com/_2t15yG-FxQA/TEov-uxpZ9I/AAAAAAAAA0I/lEqRvmQ1Bxw/s1600/1.png)

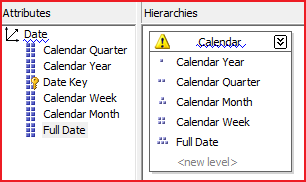
**Product Dimension:**

Drag and drop EnglishProductCategoryName from ProductCategory table, EnglishProductSubcategoryName from ProductSubcategory table and Color, ModelName, Size and Weight from Product table.

## Creating Dimension Hierarchies

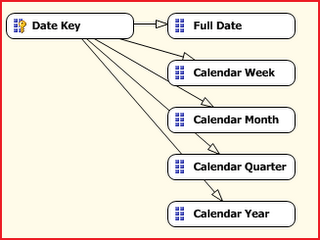
Navigate to Date Dimension Structure. Drag and drop Calendar Year attribute into **Hierarchies** surface area following by Calendar Quarter, Calendar Month, Calendar Week ,and Full Date attributes.

Rename hierarchy with **Calendar**. You'll see a warning mark in the hierarchy because attribute relationship is not set properly.

[](http://1.bp.blogspot.com/_2t15yG-FxQA/TEozbz9JewI/AAAAAAAAA0Q/6lfOm8ZYnTo/s1600/1.png)

**Set Attribute Relationships**

Click on the Attribute Relationships tab in the dimension designer. This tab is available only in Analysis Services 2008. By default, all attributes relate directly to the key attribute, Date Key as shown below:

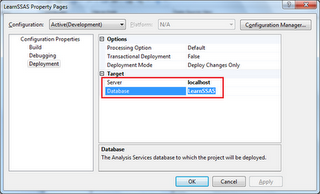
[](http://1.bp.blogspot.com/_2t15yG-FxQA/TEo3Xp32V2I/AAAAAAAAA0Y/oNpvQtmblyM/s1600/1.png)

To optimize the design by reassigning relationships, Right Click on Full Date and select **New Attribute Relationship**. Select Related Attribute as Calendar Week and Relationship type as Rigid (will not change over time). Repeat same thing for remaining attributes. Finally Attribute Relation will look like below image:

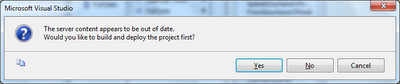
[http://2.bp.blogspot.com/_2t15yG-FxQA/TEo6kwfCqfI/AAAAAAAAA0g/USvF0CeriUA/s320/1.png](http://2.bp.blogspot.com/_2t15yG-FxQA/TEo6kwfCqfI/AAAAAAAAA0g/USvF0CeriUA/s1600/1.png)

## Deploying Cube Database from BIDS

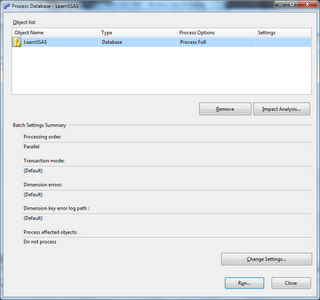
Now it’s time to deploy the cube at required server. Right click on the project (LearnSSAS in this example) and click properties to open project properties page. Enter Server Name name**Server** property and Database name in **Database** property as shown below:

[](http://1.bp.blogspot.com/_2t15yG-FxQA/TFXjO-CqYFI/AAAAAAAAA2Y/YSjINcxjgNE/s1600/1.png)

Click **OK** to save changes. Now right click on the project and click on **Process** to Build and Deploy the project. You will a message while deploying the database first time.

[](http://3.bp.blogspot.com/_2t15yG-FxQA/TFXkI5FwjQI/AAAAAAAAA2g/QBvlfTuqIMY/s1600/1.png)

Click on **Yes** to proceed. Now you can see **Process Database - PreojectName** window. Click on Run to continue.

[](http://4.bp.blogspot.com/_2t15yG-FxQA/TFXlCGEk2hI/AAAAAAAAA2o/XS6KTzg85AQ/s1600/1.png)

Once the database is deployed and processed successfully, you can see the data through **Browser** tab or directly through SQL Server Analysis Services. You can also pull cube data in Excel using Excel OLAP Pivot Tables, which is the preferred option used by business managers.

# 

# Structure of Cube

In Cube Designer, you can view and edit various properties of a cube. The designer contains the following tabs, which display different views of the cube.

## Cube Structure

Use this tab to modify the architecture of a cube.

## Dimension Usage

Use this tab to define the relationships between dimensions and measure groups, and the granularity of each dimension within each measure group. If you use multiple fact tables, you might have to identify whether measures do not apply to one or more dimensions. Each cell represents a potential relationship between the intersecting measure group and dimension.

## Calculations

Use this tab to examine calculations that are defined for the cube, to define new calculations for the whole cube or for a subcube, to reorder existing calculations, and to debug calculations step by step by using breakpoints. Calculations let you define new members and measures based on existing values, such as a profit calculation, and to define named sets.

## KPIs

Use this tab to create, edit, and modify the Key Performance Indicators (KPIs) in a cube. KPIs enable the designer to quickly determine useful information about a value, such as whether the defined value exceeds a goal or falls short of the goal, or whether the trend for the defined value is getting better or worse.

## Actions

Use this tab to create or modify drillthrough, reporting, and other actions for the selected cube. Actions provide to client applications context-sensitive information, commands, and reports that end users can access.

## Partitions

Use this tab to create and manage the partitions for a cube. Partitions let you store sections of a cube in different locations with different properties, such as aggregation definitions.

## Perspectives

Use this tab to create and manage the perspectives in a cube. A perspective is a defined subset of a cube, and is used to reduce the perceived complexity of a cube to the business user.

## Translations

Use this tab to create and manage translated names for cube objects, such as month or product names.

## Browser

Use this tab to view data in the cube.

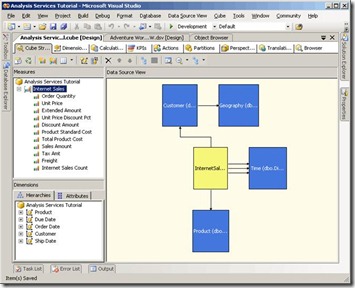
# Review Cube and Dimension properties in Cube Designer

## 1. Measure Pane

In the **Measures** pane of the **Cube Structure** tab in Cube Designer, expand the **Internet Sales** measure group.

The measures that are defined for the Internet Sales measure group appear. You can change the order of these measures by dragging the measures into the order that you want. The order will affect how certain client applications order these measures. The measure group is named Internet Sales because the underlying fact table had the friendly name of InternetSales in the data source view. Notice that a space was added automatically, based on the capitalized letter "S", to increase the user-friendliness of the name. The measure group and each measure that it contains have properties that you can edit in the Properties window.

The following image shows the measure group and measures in the **Measures** pane of Cube Designer.

[](http://bimentalist.files.wordpress.com/2011/08/clip_image002.jpg)

## 2. Dimension Pane

In the **Dimensions** pane of the **Cube Structure** tab in Cube Designer, review the cube dimensions that are in the Analysis Services Tutorial cube.

Notice that while only three dimensions were created at the database level, as displayed in Solution Explorer, there are five cube dimensions in the Analysis Services Tutorial cube. The cube contains more dimensions than the database because the Time database dimension is used as the basis for three separate time-related cube dimensions, based on different time-related facts in the fact table. These time-related dimensions are also called *role playing dimensions*. The three time-related cube dimensions let users dimension the cube by three separate facts that are related to each product sale:

The product order date, the due date for fulfillment of the order, and the ship date for the order.

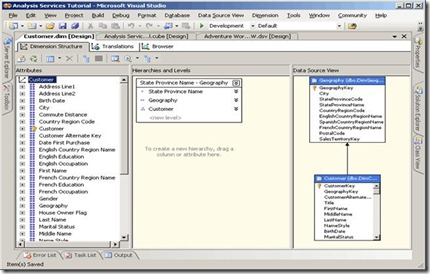
By reusing a single database dimension for multiple cube dimensions, Analysis Services simplifies dimension management, uses less disk space, and reduces overall processing time.

3. In the **Dimensions** pane of the **Cube Structure** tab, expand **Customer**, and then click **Edit Customer**.

The Customer dimension appears in Dimension Designer. (Note that Data Source View Designer and Cube Designer remain open.) Dimension Designer contains three tabs: **Dimension Structure**, **Translations**, and **Browser**. Notice that the **Dimension Structure** tab includes three panes: **Attributes**, **Hierarchies and Levels**, and **Data Source View**. The attributes that the Cube Wizard designed appear in the **Attributes** pane and the user hierarchy that the Cube Wizard defined appears in the **Hierarchies and Levels** pane. The **Data Source View** pane displays the tables in the data source view from which columns are used as attributes in this dimension.

You add, remove, and edit hierarchies, levels, and attributes on the **Dimension Structure** tab of Dimension Designer.

The following image shows the **Dimension Structure** tab of Dimension Designer.

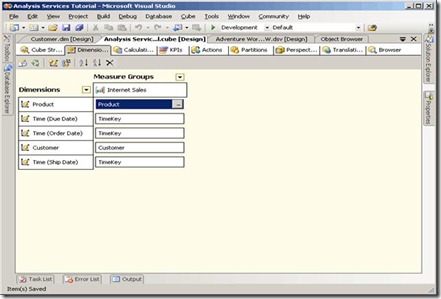


4. Switch to Cube Designer by clicking the tab in the design environment or by right-clicking the Analysis Services Tutorial cube in the Cubes node in Solution Explorer and then clicking **View Designer**.

5. In Cube Designer, click the **Dimension Usage** tab.

In this view of the Analysis Services Tutorial cube, you can see the cube dimensions that are used by the Internet Sales measure group. When a cube has multiple measure groups, cube dimensions might be used with some measure groups but not with others. Also, you define the type of relationship between each dimension and each measure group in which it is used.

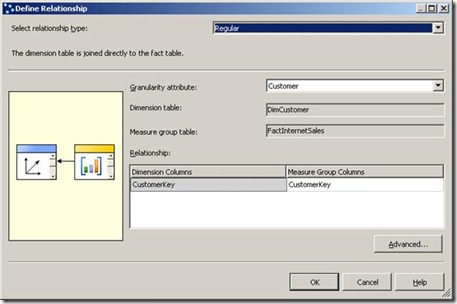
The following image shows the **Dimension Usage** tab of Cube Designer.



6. Click the **Customer** field next to **Customer** at the intersection of the Internet Sales measure group and the Customer dimension, and then click the ellipsis button (**…**).

The **Define Relationship** dialog box appears. In this dialog box, you define the custom dimension properties within a specific measure group. By default, dimensions have the same behavior in each measure group. However, they can have different behavior in different measure groups. Notice that the relationship of the Customer dimension to the Internet Sales measure group is a Regular relationship, which means that the DimCustomer dimension table is directly joined to the FactInternetSales measure group table. Notice also that the granularity of this dimension is at the lowest level, namely the Customer level, but that you can define different levels of granularity. In Lesson 5, you will learn about defining a custom granularity level.

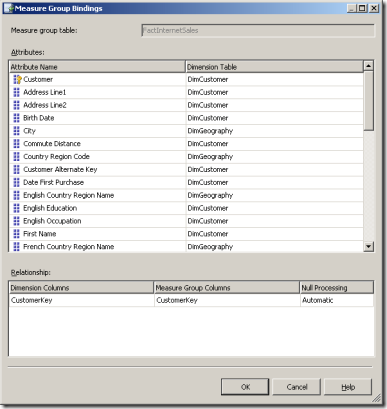
The following image shows the **Define Relationships** dialog box.



7. Click **Advanced**.

The **Measure Group Bindings** dialog box appears, which lets you change the binding of each attribute and define null processing settings. The binding for an attribute specifies the column in the underlying dimension table to which the attribute is bound. By default, this setting is inherited from the dimension; this setting is rarely changed at the measure group level. Null processing settings let you define how Analysis Services treats null values during processing at the measure group level; these settings override any settings at the dimension level.

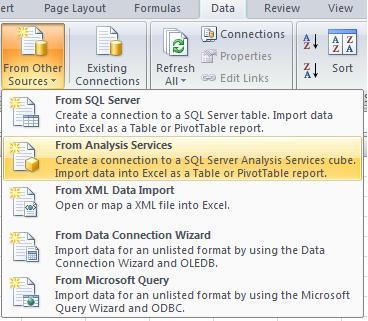
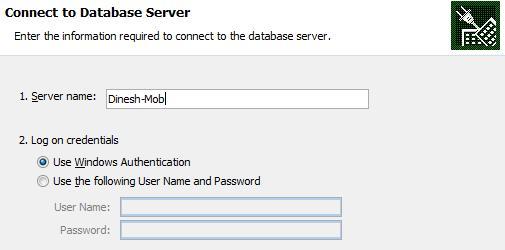
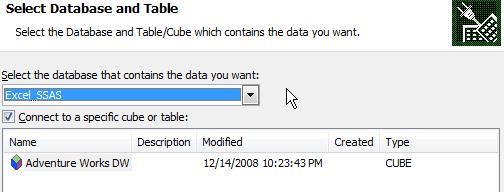
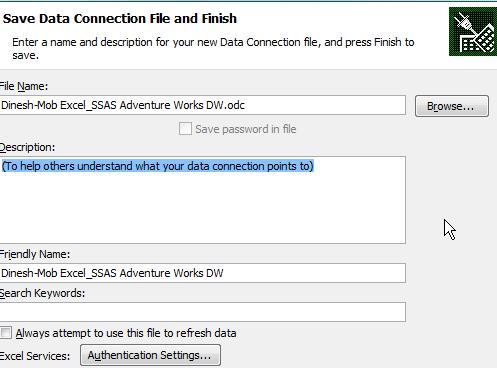
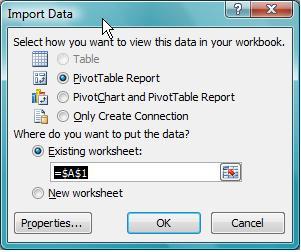
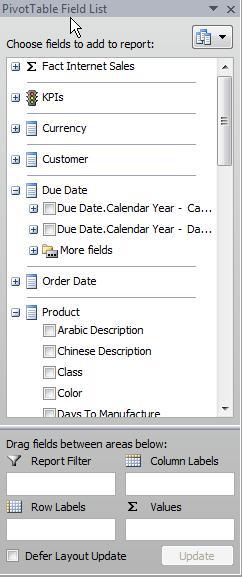
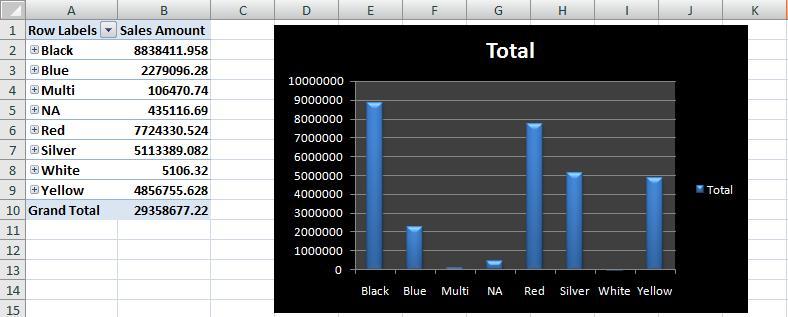
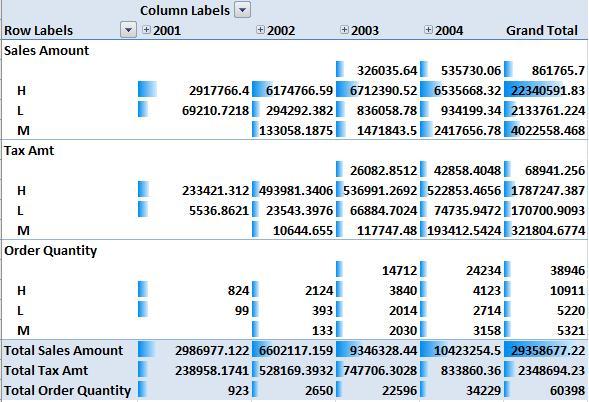
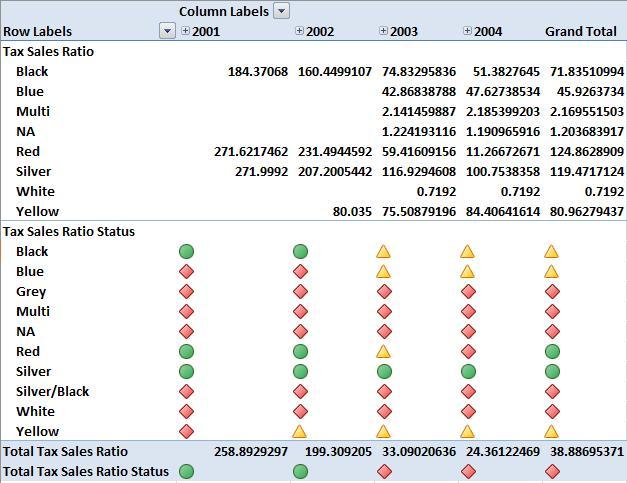
The following image shows the **Measure Group Bindings** dialog box.



8. Click **Cancel**, and then click **Cancel** again, to return to Cube Designer.

**Accessing Cubes from Excel**

Introduction  
  
If reporting access to cubes can be provided from Microsoft Excel, then report building can be performed by an end user. Majority of the time, using this method, users can construct reports the way they wish. Improvements in Excel 2007 have provided a number of new fancy features that can be used with cubes.  
  
Connecting to SSAS using Microsoft Excel

Launch Microsoft Excel first. Select the option, Select the Data table, then select the From Other Services button in the Data ribbon, then select From Analysis Services. As shown in Figure 2, a list of available data sources is provided.  
  
   
  
Figure 2: Other Sources Option  
  
In the Analysis Services option is the screen shown in Figure 3. To connect to the SSAS servers, login credentials need to be provided. However, SSAS does not support SQL Server authentication, hence the Windows Authentication option must be selected.   
  
   
  
Figure3: Login Credentials to SSAS Server.  
  
After providing the login credentials, the SSAS database and the Cube need to be selected as shown in Figure 4.  
  
   
  
Figure 4: SSAS database and Cube  
  
All cubes for the selected SSAS database will be listed. Only one cube can be specified at this point.  The next step is to set the configurations to the Data connection file. There are options to specify the file name and the path for the Data connection file.  
  
  
   
Figure 5: Data Connection File  
  
Next select (from the Figure 6) whether a Report or chart is wanted.   
  
   
  
Figure 6: How to view data.  
  
Viewing Data  
Now are the most fascinating steps, viewing the SSAS data from an Excel file.  In the right side of the excel sheet, notice a PivotTable Field List as seen in Figure 7.  Within this list the KPI, measures and dimension attributes can be seen; the user is able to select the attributes they want.  
  
   
  
Figure 7: Pivot Table Field List  
  
To see the Sales amount for each product color and class.; simply select Class and Color from the Product dimension and Sales Amount.  
  
  
   
Figure 8: Pivot Table  
  
A pivot table can be designed with columns and row headers as shown in Figure 9.  
  
  
   
Figure 9: Pivot table with Conditional Formatting  
  
Figure 9 showing a pivot table with the Conditional formatting available in Excel 2007. As this is a Excel sheet, all the additional features available with Excel are provided.   
  
KPIs are another important feature in SSAS. As this is not the place to describe how to create KPIs using SSAS, the discussion will only include how to display KPI values in a Excel sheet.  
  
The available KPI’s will be listed in the Pivot Table Filed List. For each KPI there will be three attributes, Value, Status and Trend.   
  
  
   
Figure 10 : KPIs in Pivot Table Field List  
  
After selecting the appropriate KPIs and it’s attributes, users can be given information akin to what is shown below in Figure 11.  
  
  
   
Figure 11: KPI Values

# Introduction to MDX

MDX is a language that allows describing multidimensional queries.

As a matter of fact, the name MDX stands for **M**ulti-**D**imensional e**X**tensions. As this name suggests, MDX is an extension of SQL, so you will find its structure to be similar to this of SQL, and the same keywords serving similar functions. There will be some differences however.

MDX builds multidimensional view of the data when SQL builds relational view.

The following table helps to draw analogies between relational and multidimensional terms and it is suggested, that you’ll refer to this table when MDX and SQL queries are compared

|  |  |
| --- | --- |
| Multidimensional term | Relational analog |
| Cube | Table |
| Level | Column (string or discrete number) |
| Dimension | Several related columns or dimension table |
| Measure | Column (discrete or continuous numeric) |
| Dimension member | The value in the specific row and column of dimension table |

# Sample cube

All examples in the following sections will be built around sample cube. This cube is sales analysis cube for hypothetical company which sells computer accessories world-wide. This cube contains typical dimensions such as Time, Geographic, Products, and Customers etc.

# Understanding MDX query

Let’s start to build our first MDX query. The basic MDX query has the following structure:

SELECT *axis1* ON COLUMNS, *axis2* ON ROWS FROM *cube*

*Select measures on columns, dimensions on rows from cube*

*Select dimensions on columns, measures on rows from cube*

Let’s compare that to the similar SQL statement:

SELECT column1, column2, …,columnnFROM table

### MDX & SQL QUERY

You may find some of MDX keywords familiar. Indeed, SELECT and FROM serve exactly the same purpose they do in SQL. The SELECT keyword tells us that we want to execute a query and get data results back. The FROM keyword specifies the source of the data. In the SQL case we receive the data from the table, and in MDX case we receive data from the cube.

However, there is something different between above two queries. SQL basically gives us relational view of the data, which is always two-dimensional. The result of the SQL query is a table which has two dimensions: rows and columns. And these dimensions are not symmetrical. Rows are all of the same structure, and this structure is defined by the columns. Columns, of course may be of different types and have different meanings.

In the MDX world, we can specify any number of dimensions to form result of our query. We will call them *axes* to prevent the confusion with cube’s dimensions. We can have zero, one, two, three or any other number of axes. And all these axes are perfectly symmetric.

There is no special meaning to rows and to columns in MDX other than for display purposes. You should note however, that most OLAP tools (such as Microsoft’s MDX Sample) are not capable in visualizing of MDX query results containing more than two axes. The list of columns in SQL defines the structure of the result data. In MDX, you have to define the structure of every axis. Therefore in our example *axis1* will be definition of the axis displayed horizontally, and *axis2* will be definition of the axis displayed vertically.

Axis is a collection of dimension members, or more generally *tuples*. We will discuss tuples later, so for now we’ll restrict ourselves to the axes containing members only.

There are many versatile ways to define an axis in MDX. We will start with simple one.

Suppose we want to present on the axis all members of certain dimension. The syntax for such an axis definition would be

**<dimension name>.MEMBERS**

And similarly, if we want to see all dimension members belonging to the certain level of this dimension, the syntax would be

**<dimension name>.<level name>.MEMBERS**

Now let’s construct our first MDX query using all information we learned so far. We’ll use as an example cube ‘Sales’. (The definition of this cube can be found in appendix). Our first MDX query will show sales in different continents over years. Therefore the query will look like

**SELECT**

**Years.MEMBERS ON COLUMNS,**

**Regions.Continent.MEMBERS ON ROWS**

**FROM Sales**

The result of this query visualized through the table will look like the following:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1994 | 1995 | 1996 | 1997 |
| N. America | 120,000 | 200,000 | 400,000 | 600,000 |
| S. America | - | 10,000 | 30,000 | 70,000 |
| Europe | 55,000 | 95,000 | 160,000 | 310,000 |
| Asia | 30,000 | 80,000 | 220,000 | 200,000 |

Let’s try to analyze the results of this query. We have a table with two axes. Horizontal axis (i.e. columns) contains members of dimension ‘Years’ and vertical axis (i.e. rows) contains members of level ‘Continent’ of dimension ‘Regions’. All these are explicitly specified in the query. However, one can ask: “And what are numbers in this table mean?”. Indeed, if you look into the query, you will find no reference to any quantitative data. So what happened? We know that in the cube we have to always specify measures, which are our quantitative data. Since, we didn’t explicitly tell in our MDX query which measure we want to use, the *default* measure was used. In our example, he default measure was ‘Sales’ (not to be confused with cube name ‘Sales’!). So, for instance, the interpretation of leftmost top number in the table is “In the year 1994, the sales in N.America were 120,000”. Now for comparison, let’s construct SQL query which produces the same data.

# Cells, Tuples, and Sets

As SQL returns a subset of two-dimensional data from tables.

MDX returns a subset of multidimensional data from cubes.

The cube diagram illustrates that the intersection of multidimensional members creates cells from which you can obtain data. To identify and extract such data, whether it be a single cell or a block of cells, MDX uses a reference system called tuples. Tuples list dimensions and members to identify individual cells as well as larger sections of cells in the cube; because each cell is an intersection of all the dimensions of the cube, tuples can uniquely identify every cell in the cube. For the purposes of reference, measures in a cube are treated as a private dimension, named Measures, in the cube itself. For example, in the preceding diagram, the following tuple identifies a cell in which the value is 240:

(Source.[Eastern Hemisphere].Africa, Time.[2nd half].[4th quarter], Route.Air, Measures.Packages)

# Tuple

The tuple uniquely identifies a section in the cube; it does not have to refer to a specific cell, nor does it have to encompass all of the dimensions in a cube. The following examples are all tuples of the cube diagram:

(Source.[Eastern Hemisphere])

(Time.[2nd half], Source.[Western Hemisphere])

These tuples provide sections of the cube, called slices, that encompass more than one cell.

An ordered collection of tuples is referred to as a set. In an MDX query, axis and slicer dimensions are composed of such sets of tuples. The following example is a description of a set of tuples in the cube in the diagram:

{ (Time.[1st half].[1st quarter]), Time.[2nd half].[3rd quarter]) }

In addition, it is possible to create a named set. A named set is a set with an alias, used to make your MDX query easier to understand and, if it is particularly complex, easier to process.

# Axis and Slicer Dimensions

In SQL, it is usually necessary to restrict the amount of data returned from a query on a table. For example, you may want to see only two fields of a table with forty fields, and you want to see them only if a third field meets specific criteria. You can accomplish this by specifying columns in the SELECT statement, using a WHERE statement to restrict the rows that are returned based on specific criteria.

In MDX, those concepts also apply. A SELECT statement is used to select the dimensions and members to be returned, referred to as axis dimensions. The WHERE statement is used to restrict the returned data to specific dimension and member criteria, referred to as a slicer dimension. An axis dimension is expected to return data for multiple members, while a slicer dimension is expected to return data for a single member.

The terms "axis dimension" and "slicer dimension" are used to differentiate the dimensions of the cells in the source cube of the query, indicated in the FROM clause, from the dimensions of the cells in the result cube, which can be composed of multiple cube dimensions.

# MDX QUERY SAMPLES

SELECT [Measures].[Fact Sales Count] ON COLUMNS,

[Dim Car].[Car Id].[Car Id] ONROWSFROM [SSASDB 1]

SELECT [Measures].[Sales Amount] ONCOLUMNS,

[DimProduct].[English Product Name].MEMBERS ON ROWS FROM [AdventureWorks\_Cube]

SELECT [Measures].[Fact Sales Count] ON COLUMNS,

[Dim Car].[Car Id].[&Maruti] ONROWSFROM [SSASDB 1]

SELECT [Measures].[Fact Sales Count] ONCOLUMNS

FROM [SSASDB 1]

SELECT [Measures].[Revenue] ONCOLUMNS

FROM [SSASDB 1]

SELECT [Measures].[Revenue] ONCOLUMNS,

[Dim Car].[Car Type] ONROWS

FROM [SSASDB 1]

--output

--All 65784000

SELECT [Measures].[Revenue] ONCOLUMNS,

[Dim Car].[Car Type].MEMBERSONROWS

FROM [SSASDB 1]

-- Revenue

--All 65784000

--HATCHBACK 15184000

--LUXURY 33500000

--SEDAN 17100000

--Unknown (null) -- Unknown added by default

***NONEMPTY FUNCTIONS***

SELECT Measures.Revenue ONCOLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ONROWS

FROM [SSASDB 1]

***--NONEMPTY FUNCTIONS RETURNS THE TUPLES THAT ARE NOT EMPTY***

SELECT Measures.MEMBERSONCOLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ONROWS

FROM [SSASDB 1]

SELECT [Measures].MEMBERSONCOLUMNS,

NONEMPTY([DimProduct].[English Product Name].MEMBERS) ONROWS

FROM [AdventureWorks\_Cube]

SELECT [Measures].[Sales Amount] ONCOLUMNS,

[DimProduct].[English Product Name].MEMBERSONROWS

FROM [AdventureWorks\_Cube]

***--MEMBERS FUNCTIONS CAN BE USED WITH MEMBERS DIMENSIONS,***

***--RETURNS ONLY MEASURES DEFINED DIRECTLY WITHIN THE CODE***

Revenue Fact Sales Count

All 65784000 90

HATCHBACK 15184000 37

LUXURY 33500000 25

SEDAN 17100000 28

SELECT

ADDCALCULATEDMEMBERS(Measures.MEMBERS) ONCOLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ONROWS

FROM [SSASDB 1]

--addcalculatedmembers returns all measures , even calculated members

SELECT

Measures.ALLMEMBERS ON COLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ON ROWS

FROM [SSASDB 1]

--ALLMEMBERS returns all measures

**Add selected members in columns:**

SELECT {Measures.[Fact Sales Count],Measures.[Revenue]} ON COLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ON ROWS

FROM [SSASDB 1]

SELECT {Measures.[Sales Amount],Measures.[Freight]} ONCOLUMNS,

[DimProduct].[English Product Name].MEMBERS ON ROWS

FROM [AdventureWorksDW2008R2\_Cube]

**Add selected members in ROWS:**

SELECT {[Measures].[Fact Sales Count],[Measures].[Revenue]} ONCOLUMNS,{[Dim\_Car].[Car Name].ALLMEMBERS \* [Dim\_Customer].[First Name].ALLMEMBERS} ON ROWS FROM [CAR\_CUBE]

SELECT {Measures.[Sales Amount], Measures.[Freight]} ONCOLUMNS,

{[DimProduct].[English Product Name].MEMBERS \* [DimCustomer].[First Name].MEMBERS}

ONROWS

FROM [AdventureWorksDW2008R2\_Cube]

**Formatting Data in MDX**

WITHMEMBER Measures.[TotalRevenueFormmated] AS([Measures].[Revenue]),FORMAT\_STRING="#,##,#.00"

SELECTNONEMPTY {Measures.[TotalRevenueFormmated],

[Measures].[Fact Sales Count]} ONCOLUMNS,

NONEMPTY [Dim Car].[Car Type].MEMBERSonROWS

FROM [SSASDB 1]

**WHERE CLAUSE**

Now let us use the WHERE clause. The WHERE clause determines which dimension or member is to be used as a slicer. A query can have multiple axes, but when a query has 3 axes we will not be able to display it (in SSMS). For this the WHERE clause has been provided.

WHERE CLAUSE

SELECT {Measures.[Fact Sales Count],Measures.[Revenue]} ONCOLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ONROWS

FROM [SSASDB 1] WHERE[Dim Customer].[CID].&[C10]

SELECT {Measures.[Sales Amount],Measures.[Freight]} ONCOLUMNS,{[DimProduct].[English Product Name].MEMBERS \* [DimCustomer].[First Name].MEMBERS}

ONROWS

FROM [AdventureWorksDW2008R2\_Cube]

WHERE [DimCustomer].[Gender].&[M]

SELECT {Measures.[Fact Sales Count],Measures.[Revenue]} ONCOLUMNS,

NONEMPTY([Dim Car].[Car Type].MEMBERS) ONROWS

FROM [SSASDB 1] WHERE([Dim Customer].[CID].&[C10],[Dim Sales Person].[Sales Person ID].&[SP11])

SELECT {Measures.[Sales Amount],Measures.[Freight]} ONCOLUMNS,

{[DimProduct].[English Product Name].MEMBERS \* [DimCustomer].[First Name].MEMBERS}

ONROWS

FROM [AdventureWorksDW2008R2\_Cube]

WHERE ([DimCustomer].[Gender].&[M],[DimCustomer].[House Owner Flag].&[0])

**SUM CLAUSE:**

WITH MEMBER Measures.TotalAS SUM  
( { [Order Date].[Calendar].[Calendar Year].&[2001]  
, [Order Date].[Calendar].[Calendar Year].&[2003]},  
[Measures].[Sales Amount])  
SELECT Measures.TotalON0  
,NON EMPTY [Product].[Product-Category].[Product Category Key].Members ON 1  
FROM DemoCube

**MAX CLAUSE :**

WITH

MEMBER Measures.[Max Value] AS Max  
(

[Order Date].[Calendar].[Calendar Quarter],

[Measures].[Sales Amount]  
)  
SELECT Measures.[Max Value] ON 0,  
NON EMPTY [Order Date].[Calendar].[Calendar Quarter] \*  
[Product].[Product-Category].[Product Category Key].MEMBERS  
ON 1FROM DemoCube

Max can be replaced by min if required.

**Calculation SubTab:**

Adding Calculated Members.

WITH MEMBER Measures.[TotalRevenue] AS

([Measures].[Revenue] \* [Measures].[Fact Sales Count])

SELECTNONEMPTY {Measures.[TotalRevenue],

[Measures].[Fact Sales Count]} ONCOLUMNS,

NONEMPTY [Dim Car].[Car Type].MEMBERSonROWS

FROM [SSASDB 1]

**Getting UserName :**

WITH MEMBER Measures.[Name] AS UserName  
SELECT Measures.[Name] ON COLUMNS  
FROM DemoCube

CREATE MEMBER CURRENTCUBE.Measures.ProfitRatio AS 'Measures.[Store Sales]/Measures.[Store Cost]', SOLVE\_ORDER = 10

# FUNCTIONS

This is the third and final part of the series and we will explore some additional functions of MDX:

## COUNT FUNCTION

Let us discuss different ways of using the COUNT function. The following query returns the number of hierarchies in a cube:

WITH MEMBER MEASURES.[Count] ASdimensions.count  
SELECT Measures.[Count] ON 0FROM DemoCube

WITHMEMBER MEASURES.[Count] ASdimensions.count

SELECT Measures.[Count] ON 0FROM [AdventureWorksDW2008R2\_DV]

The following query returns number of levels in the product categories:

WITH MEMBER measures.[Count] AS[Product].[Product-Category].Levels.Count  
Select Measures.[Count] ON 0FROM DemoCube

The following query counts the number of cells in a set of members that consists of children of a category:

WITH MEMBER measures.XAS  
[Product].[Product Category Key].children.count  
SELECT Measures.XON0  
FROM DemoCube

We can exclude non empty cells in the following manner:

WITH MEMBER measures.[Count] AS  
Count([Product].[Product Category Key].children, EXCLUDEEMPTY)  
SELECT Measures.[Count] ON 0  
FROM DemoCube

2. IIF functions  
returns one of the possible two values in a logical test. It can return either a string or numeric value.

WITH MEMBER MEASURES.[IIF Value] AS  
IIF([Measures].[Order Quantity]>1000  
, "Qty High", "Qty Low")  
SELECT {[Measures].[Order Quantity],MEASURES.[IIF Value]} ON 0,  
[Order Date].[Calendar].[Calendar Quarter].MEMBERS ON 1  
FROM DemoCube

WITHMEMBER MEASURES.[IIF Value] AS

IIF([Measures].[Order Quantity]>1000

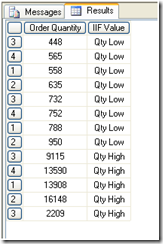
, "Qty High", "Qty Low")

SELECT {[Measures].[Order Quantity],MEASURES.[IIF Value]} ON 0,

[Order Date].[Calendar Year].MEMBERSON 1

FROM [AdventureWorksDW2008R2\_DV]

In this query, we are displaying whether the quantity is low or high, according to the value for Order Quantity.

[](http://lh6.ggpht.com/_toA00pxrCRM/SqtYTkh84TI/AAAAAAAAAHQ/WnhMd1bITE4/s1600-h/image%5b2%5d.png)  
We can change the condition if required.

**Let us use IIF with string data**

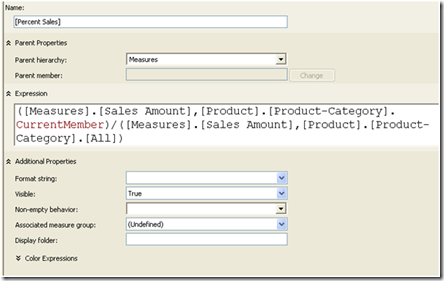
WITH MEMBER [IIF String] AS  
IIF([Product].[Product-Category]=Bikes  
, "Common", "not common")  
SELECT {[Measures].[Sales Amount],[IIF String]} ON 0,  
[Product].[Product-Category].[Product Category Key].MEMBERS on 1  
FROM DemoCube

# CALCULATED MEMBERS

Let us add a calculated member. The following query creates percent by using MDX and displays results:

WITH MEMBER Measures.[Percent] AS  
([Measures].[Sales Amount],[Product].[Product-Category].CurrentMember)  
/([Measures].[Sales Amount],[Product].[Product-Category].[All]),  
FORMAT\_STRING="Percent"  
SELECT Measures.[Percent] on 0,  
NON EMPTY [Product].[Product-Category].[Product Category Key].members on 1  
from democube

This calculation can be kept for further usage if it is added from BIDS (Business Intelligence Development Studio) as follows

  
  
Here we can give the format string to percent, if required. Later it can directly be used in MDX as follows:

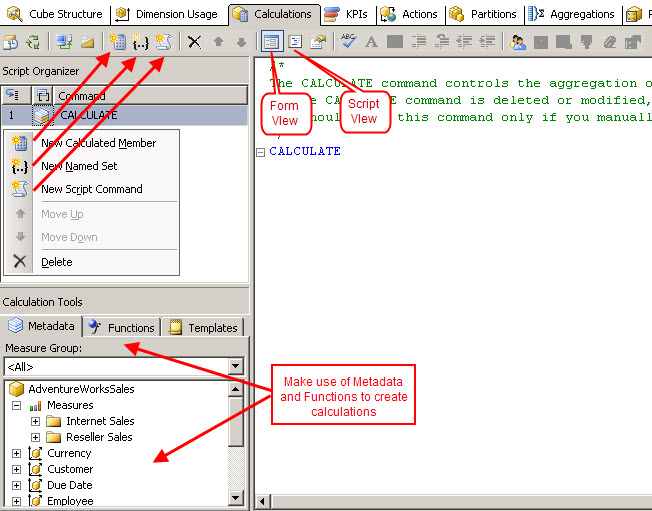
SELECT [Measures].[Percent Sales] ON0,  
NON EMPTY [Product].[Product-Category].[Product Category Key].members on1  
fromdemocube

This will give the same result as the previous query, with the difference that we already have a calculated field.

# CALCULATED MEMBERS

Calculations in Analysis Services help to extend the cube capabilities to make the BI applications more powerful. A calculation is nothing but an MDX expression or script which is used to define a calculated member, a named set, or a scoped assignment in a cube. Using calculations, we can, not only add objects defined by the cube data but can also refer other parts of the cube, other cubes, or even information outside the Analysis Services database using expressions.

To add or edit calculations, use the Calculations tab in the cube designer in BIDS. Have a look at the following screenshot which shows the various panes inside the Calculations tab.



        ﻿  The tab has three panes which are Script Organizer, Calculation Tools and Calculation Expressions which supports a form view and a script view. We can create a Calculated Member, Named Set or Script Command using the Calculations tab either by right clicking in the Script Organizer pane or by using the buttons provided in the toolbar, as shown in the above screenshot. The calculations tab supports two different views when viewing or editing calculations,

* Form View – Provides form editors to view and edit calculations in an organized and user friendly manner. Using the metadata, functions and available templates, we can easily create calculations even with limited knowledge of actual script commands.
* Script View – Is meant for more advance users who can directly write the script to create required calculations. It displays the entire MDX script associated with the cube as well as displaying the metadata, functions, and tools available to the cube.

(This is all about the designer. In this part I am going to show a simple example of creating a calculated member using the AdventureWorksDW2008 database. I have created a sample cube using Internet Sales and Reseller Sales as measure groups and some of the selected dimensions for this example.)

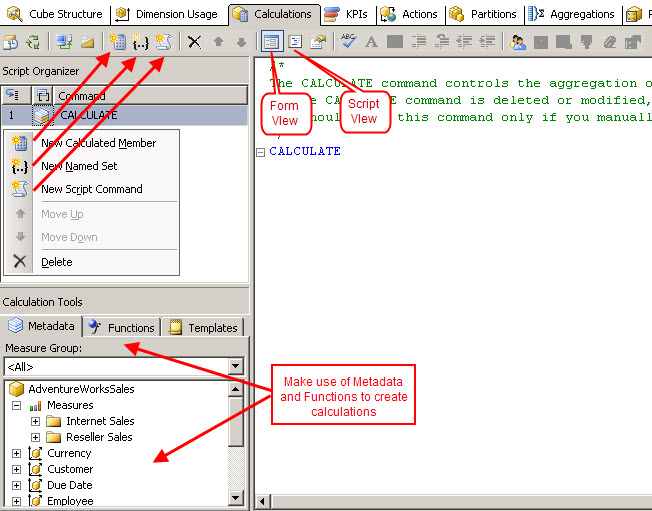
# Calculated Members

          A calculated member is defined using an MDX expression, based on which its value is calculated during run time. The cube only stores the definition of a calculated member and not a value hence avoids increasing the cube size by adding calculated members to the cube. The value only gets calculated in memory whenever required to answer the query.

          Calculated members can be defined for dimensions or measures. The members defined for measure dimensions are called as calculated measures. Calculated members are defined using the data available in the cube and can be made complex by combining data with arithmetic operators, numbers, and functions.

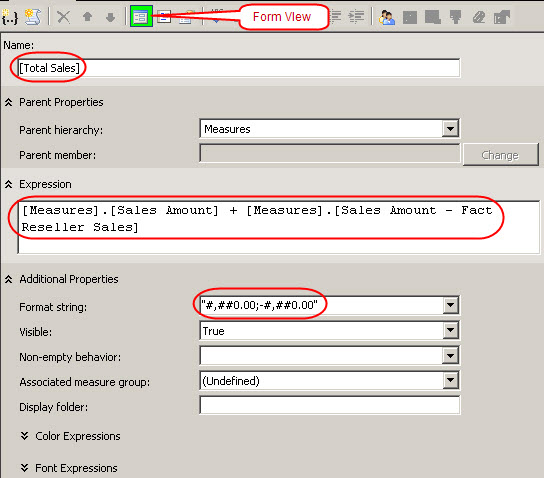
# CREATING A CALCULATED MEMBER:

**In the Calculation Tab, Click on New Calculated Member.**

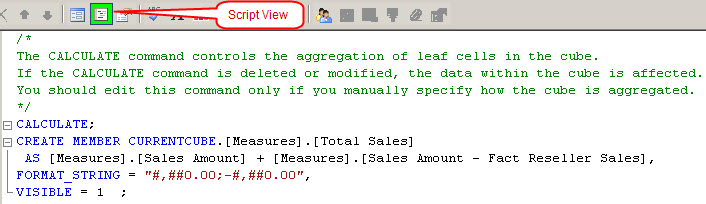


**Once clicked on New Calculated Member, a screen appears as shown below:**

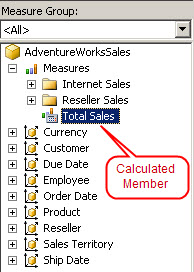
**Enter the Name, Expression as shown below.**



          The above picture shows the calculated measure called *Total Sales* in the form view which is created by adding Sales Amount measure from Internet Sales and Reseller Sales measure groups using MDX expression. We can also specify a suitable format for this new calculation. The same can be viewed in the script view which shows the actual script involved in creating this calculated measure.



***Once deployed the calculation created is available for browsing under the available measures as follows.***



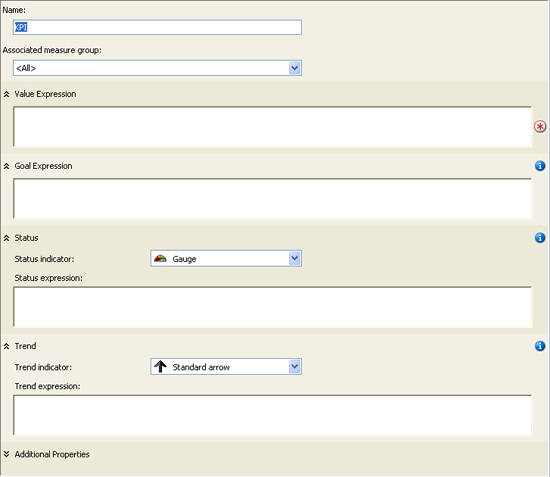
# Overview of KPI

In SQL Server Analysis Services (SSAS), you can add key performance indicators (KPIs) to your database cube in order to evaluate business performance, as reflected in the cube data. A KPI is associated with a measure group and is made up of a set of calculations. Typically, the calculations are a combination of calculated members and Multidimensional Expressions (MDX) statements.

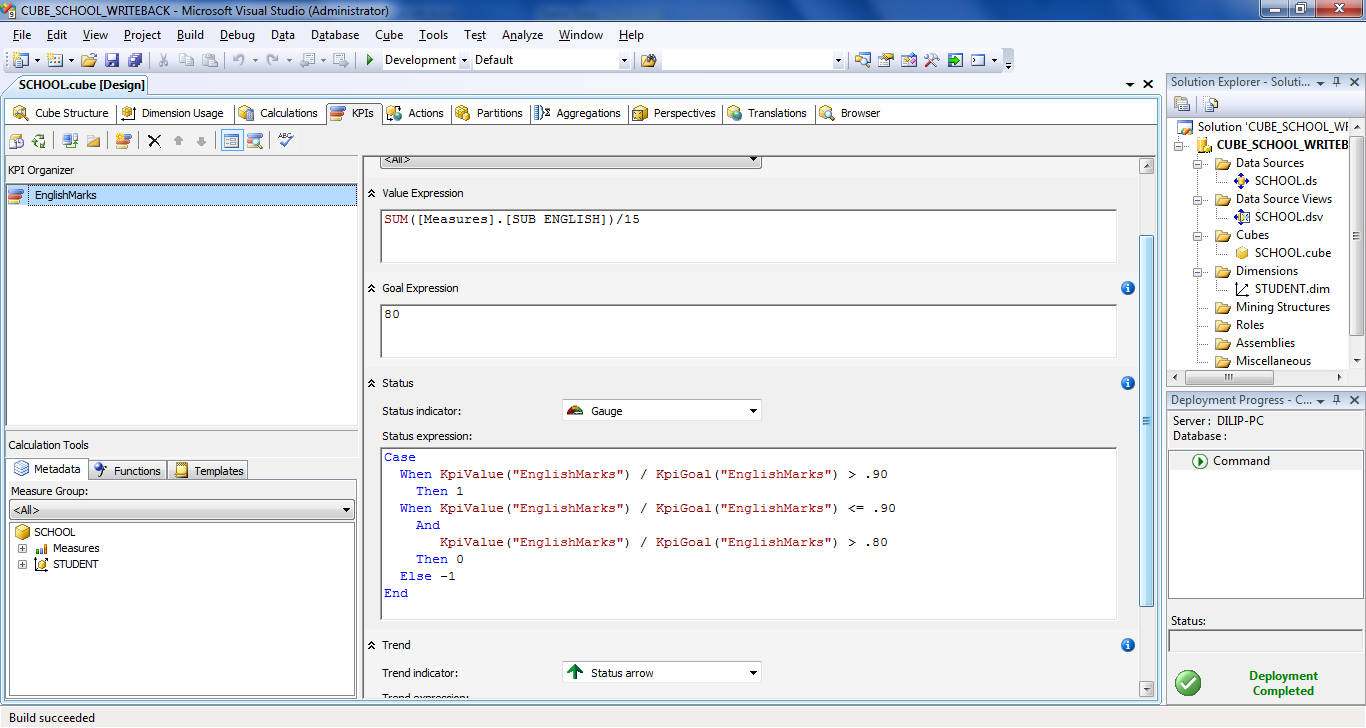
A KPI consists of four main properties that are important to evaluating business performance as shown below:

* **Value Expression.** An MDX expression that returns the KPI’s actual value.
* **Goal Expression.** An MDX expression that returns the KPI’s target value.
* **Status Expression.** An MDX expression that returns the KPI’s state at a specific point in time.
* **Trend Expression.** An MDX expression that returns the KPI’s value over time.

|  |  |
| --- | --- |
| **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. |

****

EXAMPLE 1 :



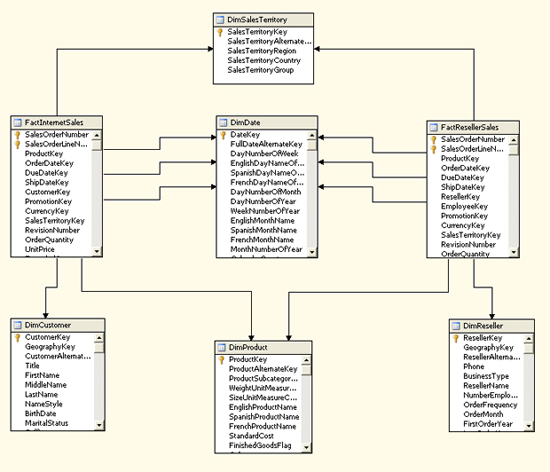
Example Using Advent

For that solution, I created the following database components:

1. A data source that points to the AdventureWorksDW2008 database on a local instance of SQL Server 2008.
2. A data source view that includes the tables shown in Figure 1.
3. Database dimensions based on each dimension table in the data source view.
4. A cube based on the database dimensions as well as on the two fact tables in the data source view.

**Measures :FactInternetSales, FactResellerSales**

**Dimensions :DimSalesTerritory,DimDate,DimCustomer,DimProduct,DimReseller**



## Creating a Calculated Member

When you create a KPI, you base one or more of your expressions on members in a measure group or dimension. However, in some cases, the existing members don’t support the type of KPI you want to create, at least not in their current form. If that’s the case, you can create a calculated member, which is similar to creating a computed column in a SQL Server database.

To create a calculated member, open your Analysis Services project in SQL Server Business Intelligence Development Studio (BIDS), and then open the cube in which you want to create your KPI. (For this article, I’m adding the KPI to the Sales cube.) In Cube Designer, click the **Calculations** tab, and then click the **New Calculated Member** button. A new calculation form opens in the right pane, as shown in Figure 2.

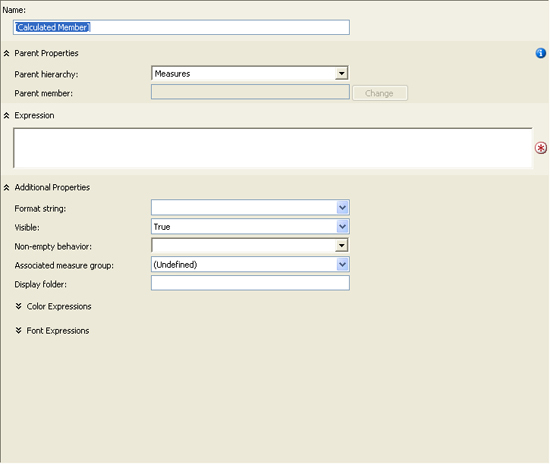


Figure 2: Form for a new calculated member

You should first name the calculated member by typing the name in the **Name** text box. For this example, I use the following name:

[Profit Margin]

Notice that I enclose the name in brackets. If your name includes a space, as mine does, you must use the brackets.

Next, you should verify the setting for the **Parent hierarchy** property. By default, the property is set to **Measures**. Because we’re creating a calculated measure, this is the hierarchy we want to use.

After you select the hierarchy, you must define an MDX expression that determines the value for your calculated measure. In this case, I want to create a measure that provides the profit margin for each sale. The following expression calculates the margin by dividing the net profit by the sales amount:

 ([Measures].[Sales Amount] -   ([Measures].[Total Product Cost] + [Measures].[Tax Amt] +

    [Measures].[Freight])) /

  [Measures].[Sales Amount]

Notice that I calculate the net profit by subtracting the total product cost, tax amount, and freight from the sales amount. (You might decide on a different formula for you net profit.) I then divide that total by the sales amount.

**Note:** When opening the **Calculations** tab, you probably noticed the list of measure groups and dimensions in the lower-left pane. You can drag a member from any of these hierarchies to your expression text box to add the *fully qualified name* of that member to the expression. Also note, the MDX expression shown above is a relatively simple one. You can, of course, create far more complex expressions. However, an in-depth discussion about MDX is beyond the scope of this article. For more information about MDX, see SQL Server Books Online.

After you create your expression for the calculated member, you can set additional properties. For this example, I set the **Format string** property to **"Percent"** and then select **Fact Internet Sales** in the **Associated Measure Group** property because I want to associate the calculated member with that measure group. Figure 3 shows what the form should look like after you’ve configured all the properties.

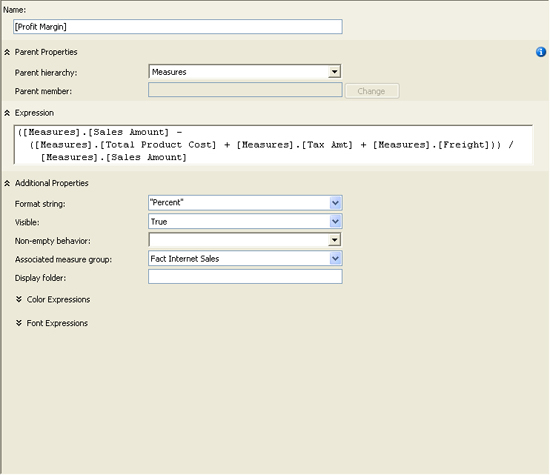


Figure 3: Creating the Profit Margin calculated member

That’s all there is to creating a calculated member. Be sure to save the project and then process the cube so the measure is available to your KPI. After you process your cube, you can verify that the measure has been successfully added by browsing the cube data and viewing the Profit Margin measure.

## Creating a Key Performance Indicator

In Cube Designer, click the **KPIs** tab, and then click the **New KPI** button. A new KPI form opens in the right pane, as shown in Fig below.

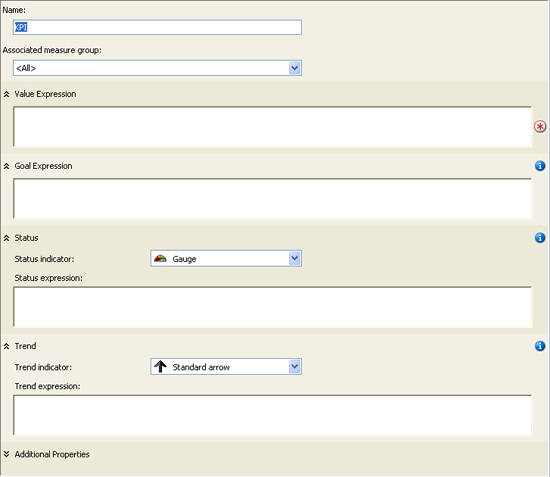


Fig: Form for a new key performance indicator (KPI)

To configure the KPI, first provide a name. (I use *Gross Profit* for our example KPI.) Then select a value for the **Associated measure group** property. (I use **Fact Internet Sales**.) You’re now ready to add the necessary expressions to your KPI.

### Adding the Value Expression

Your value expression should reflect the basic measure by which your KPI is gauged. The value returned by the expression serves as the foundation for your KPI. It’s the only one of the four expressions that’s required. For our example KPI, I use the following expression:

[Measures].[Profit Margin]

As you can see, I’m simply calling the Profit Margin calculated measure. Notice that, as you saw when creating your calculated member, I’m using the fully qualified member name. Also, as with the calculated member, you can drag the name from the hierarchies listed in the lower-left pane to the expression text box.

### Adding the Goal Expression

As the name suggestions, the goal expression indicates what your organization is trying to achieve. For example, your profit margin might currently be at 25%, but your goal might be to reach 30%. And you can also set your goal to match more specific criteria. For instance, in the example KPI, I set the goal to vary depending on the specific sales territory group, as shown in the follow MDX expression:

Case

  When [Territory].[Sales Territory Group]

    Is [Territory].[Sales Territory Group].[Europe]

      Then .34

  When [Territory].[Sales Territory Group]

    Is [Territory].[Sales Territory Group].[North America]

      Then .36

  When [Territory].[Sales Territory Group]

    Is [Territory].[Sales Territory Group].[Pacific]

      Then .32

* Else .30

End

Notice that I use a Case statement to define my criteria. The Case statement includes three When expressions, one for each territory group. Each When expression identifies the member on which to base the expression and the member value, following the Is keyword. This is followed by a Then expression which defines what action to take. For example, the first when expression states that if the sales territory group is Europe, then the profit margin goal is 34%. However, the goal for the North American group is 36%, and the goal for the Pacific group is 32%. The Else clause then specifies that all other groups have a 30% goal.

### Adding the Status Expression

Your status expression determines the current status of the KPI by comparing the goal expression to the value expression. For example, if your KPI value returns a 20% profit margin, but your goal is 30%, the status will indicate that you are below your goal. However, to arrive at the status, your status expression must return a value in the range of -1 to +1, where -1 indicates bad performance and +1 indicates good performance. For our example KPI, I use the following MDX expression to determine the status of performance:

Case

  When KpiValue("Gross Profit") / KpiGoal("Gross Profit") > .90

    Then 1

  When KpiValue("Gross Profit") / KpiGoal("Gross Profit") <= .90

    And

       KpiValue("Gross Profit") / KpiGoal("Gross Profit") > .80

    Then 0

  Else -1

End

Once again, I create a Case statement. In the first When expression, I divide the KPI value by the KPI goal and compare it to .90. If the value is greater than 90% of the goal, I assign the status value a +1. However, in the second When expression, I specify that if the KPI value is less than or equal to 90% *and* is greater than 80%, the status should be 0. Otherwise, the KPI status should be -1.

Notice that I use the KpiValue function to retrieve the KPI’s value and I use the KpiGoal function to retrieve the KPI’s goal. These functions make it easy to retrieve the value and goal within your MDX expressions.

One other thing I want to point out is the status indicator. For this example, I use the traffic light. That means, when the status value is 1, the traffic light will be green. If the status value is 0, the light will be yellow. Otherwise, the light will be red. For example, if my KPI goal is 30% and my value is 20%, the traffic light will be red. That’s because a 20% profit margin represents only about 67% my goal, which would evaluate to a -1 in the status expression. However, a value of 25% would evaluate to about 83%, which would mean a yellow light, and a value of 28% would evaluate to about 93%, which would result in a green light.

After you add the status expression and set up the status indicator, your KPI form should look similar to the one shown in Figure 5. (You have to scroll down to see the trend expression, which we’ll work on next.)

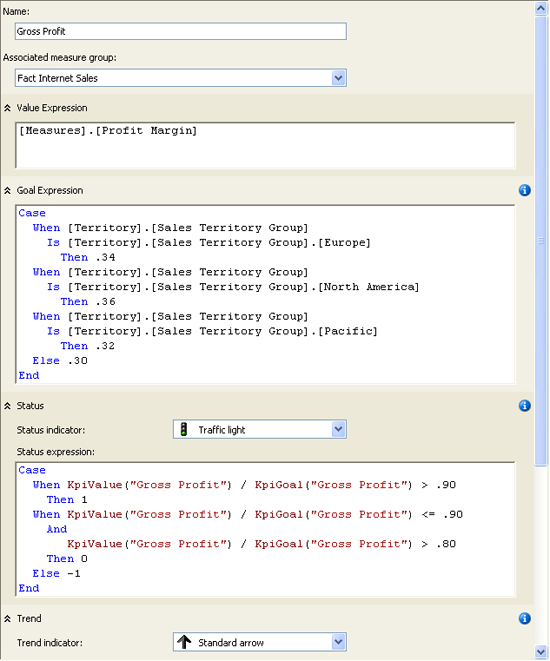


Figure 5: Creating the Gross Profit KPI

### Adding the Trend Expression

The trend expression lets you compare how your KPI is performing over time. Although the status value can tell you how well you’re achieving you goal at a fixed point in time, it doesn’t tell you how those achievements compare to another point in time. For example, your organization might have reached 92% of its goal this year, but reached 95% the year before. Although the performance looks good when just viewing this year’s total, that performance measure doesn’t reveal that this is actually a downward trend. And this is what the trend expression allows you to find out.

For the Gross Profit example KPI, I use a trend expression that compares this year’s totals to the previous year, as shown in the following Case statement:

Case

  When IsEmpty(ParallelPeriod([Order Date].[Calendar Year].[Calendar Year],

      1, [Order Date].[Calendar Year]))

    Then 0

  When [Measures].[Profit Margin] >

    (ParallelPeriod([Order Date].[Calendar Year].[Calendar Year],

      1, [Order Date].[Calendar Year]), [Measures].[Profit Margin])

    Then 1

  When [Measures].[Profit Margin] =

    (ParallelPeriod([Order Date].[Calendar Year].[Calendar Year],

      1, [Order Date].[Calendar Year]), [Measures].[Profit Margin])

    Then 0

  Else -1

End

Notice that the first when expression includes the IsEmpty function and the Parallel Period function. The IsEmpty returns a value of true if the evaluated expression an empty cell value. Otherwise the function returns false. I use this function in the first when expression to determine whether the year preceding the current one exists in the cube data. If not, then the trend expression returns a 0, which indicates a neutral trend. (A trend expression, like the status expression, should return a value from -1 to +1.)

The ParallelPeriod function returns a member value from a previous period that’s in the same relative position as the current member. Because I specify Calendar Year, the previous member will also be based on Calendar Year. The function includes three arguments. The first argument returns the level of the hierarchy that you want to target. In this case, it’s the Calendar Year level of the Calendar Year hierarchy, which is part of the Order Date hierarchy. The second argument, 1, indicates the number of units that you want to go back. Because Calendar Year is the current position, the function will go back one year. The final argument specifies that I am basing the calculation on Calendar year. As a result, the first when expression in the Case statement above will determine if the previous calendar year exists, and if it doesn’t the trend will be set to 0.

The second when expression determines whether the profit margin for the current year is greater than the profit margin for the previous year. Notice that the third argument in the Parallel Period function now specifies Profit Margin, rather than Calendar Year. If the profit margin is greater in the current year, the trend value is set to +1. However, the third When expression determines whether the profit margin for the current year *equals* that of the previous year. If so, the trend value is set to 0. Otherwise the trend value is set to -1.

Notice also that you can set the trend indicator. You have three different types of arrows from which to choose. Or you can pick the smiley face!

## Completing the Key Performance Indicator

If you click the **Additional Properties** down arrow at the bottom of the KPI form, you can view and configure the following properties:

* **Display folder.** The folder in which the KPI can be found when browsing the cube.
* **Parent KPI.**> A KPI that acts as the parent of the current KPI so the parent KPI can use the value of the child KPI.
* **Current time member.** An MDX expression that returns a member that identifies the KPI’s temporal context.
* **Weight.** An MDX expression that assigns a weight to a child KPI to indicate its relative importance in the parent KPI.
* **Description.** A description of the KPI.

After you’ve completed your KPI, you can then view its results, based on the current values in the cube data. To view the KPI, click the **Browser View** button on the **KPIs** tab. The browser view includes two panes. The top pane lets you define filters that determine what data the KPI uses, and the bottom pane displays the KPI. By default (before any filters are defined), the KPI calculates the KPI value for the entire data set. However, you can create filters that let you define the data for which you want to run the KPI.

For example, Figure 6 shows the filter I created for the Gross Profit KPI. I first selected Pacific as the sales territory group, and then selected 2004 as the calendar year. As you can see in the figure, the KPI value is 29.72%. Because the goal for the Pacific group is only 32%, the Pacific group reached nearly 93% of its goal for 2004, which means the status indicator is green. However, notice that the trend points downward. That’s because they Pacific group did better in 2003 (30.58%).

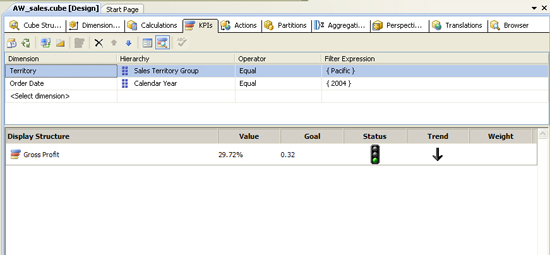


Figure 6: Browsing the Gross Profit KPI

As this example demonstrates, a KPI can be an effective way to get a quick read on your organization’s performance. Of course, how the KPI is displayed and what sort of status indicators you use will depend on the implementation of the client applications. But Analysis Services provides the features you need to make the KPI data easily available to your cube. And once you have an understanding of the basic components that make up a KPI, you can start implementing KPIs that support the various business needs of your organization.

# SSAS Actions

In SSAS, actions are one of the best ways to customize browsing and drilling of data for the end-user.

There are three types of actions in SSAS:

* Report
* Drillthrough
* Standard

 Actions are powerful way of extending the value of SSAS cubes for the end user. They can click on a cube or portion of a cube to start an application with the selected item as a parameter, or to retrieve information about the selected item. Actions haven't been well-documented until now, Robert Sheldon once more makes everything clear.

One of the objects supported by a SQL Server Analysis Services cube is the *action*. An action is an event that a user can initiate when accessing cube data. The event can take a number of forms. For example, a user might be able to view a Reporting Services report, open a Web page, or drill through to detailed information related to the cube data

In this post, we will learn to define actions in your Microsoft SQL Server Analysis Services (SSAS) project. An action is just a Multidimensional Expressions (MDX) statement that is stored in Analysis Services and which can be incorporated into client applications and started by a user.

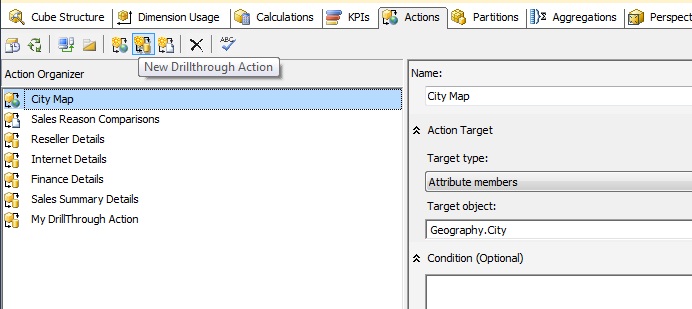
Although the basic steps necessary to implement the various types of actions are similar, the configuration of each one is unique. In this article, I explain how to add a drill through action to your cube.

# Creating a DrillDown Action

For this example, we will be using the sample Adventureworks SSAS project that ships with SQL Server 2008.

Step 1  
Open the Adventureworks cube.

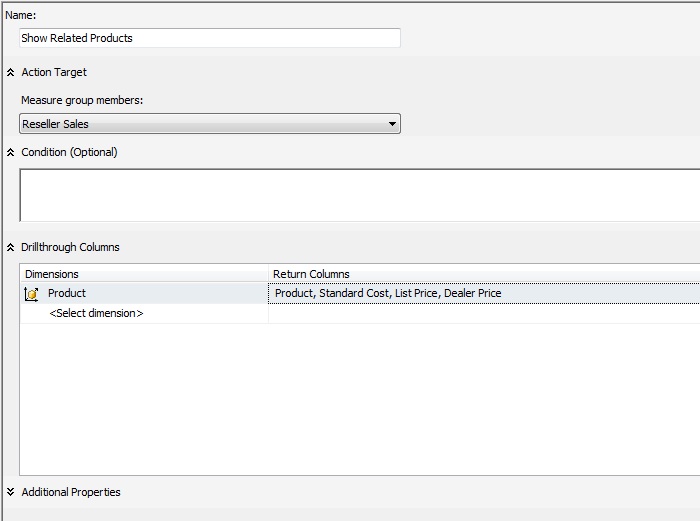
Step 2  
Open the Actions tab and from the toolbar select "**New Drill through Action**" as shown below.



Step 3  
When you create a new action, you will get a screen like below to define the action.

Let's say we want to see products related to the Reseller Sales amount.

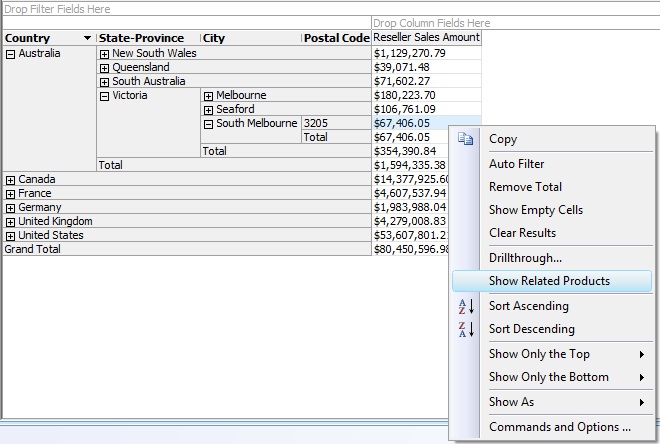
* **Name**: I named the action "Show Related Products".
* **Measure group members**: we need to select the target field that should offer the drillthrough action and in our case we selected Reseller Sales amount.
* **Condition**: to keep it simple for now, we will not use the condition property.
* **Return Columns**: select the columns to show when the user selects this action.



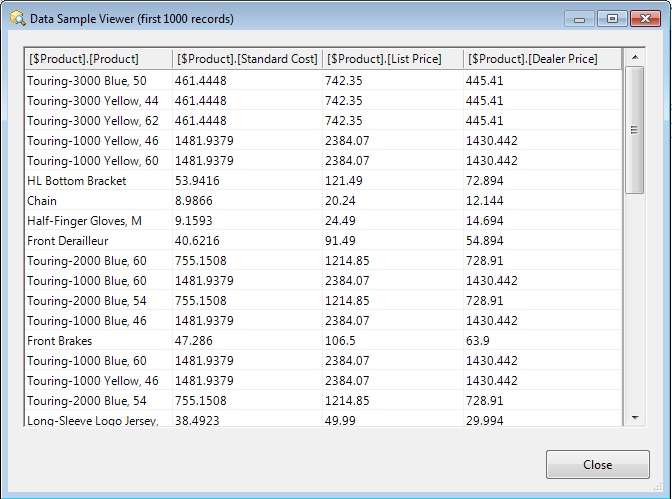
Step 4  
Deploy this change and browse the cube in the browser tab.

I selected the Reseller Sales Amount measure and sliced it by the Geography hierarchy from the Geography dimension as shown below.

Right click on any value in the Reseller Sales Amount and you should find "Show Related Products" in the menu as shown below.



Step 5  
Click on the Show Related Products menu option and you will get a pop-up window with the fields we defined in the action showing all the records that make up the value on which we selected the drill through action.



# Creating a Reporting Action

**Steps to create the Reporting Action**

1. From BIDS, open the Analysis Service cube project. Navigate to Actions. Right Click on Action Organizer and click New Reporting Action (Shown in Figure 4).

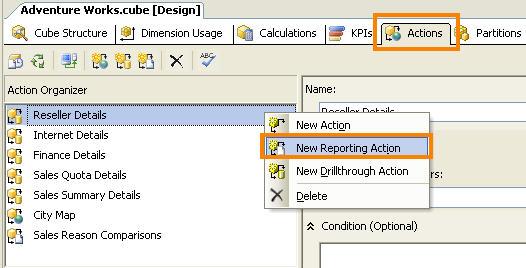


Figure 4 - Create New Reporting Action

2. Select Target Type as Cells. For this example, Cells target type would be the best choice as we have to specify more than one dimension as conditional expression in the next step. (Figure 5)

3. Specify the path of the SQL Server Report in **URL Access format** as shown in Figure 5.

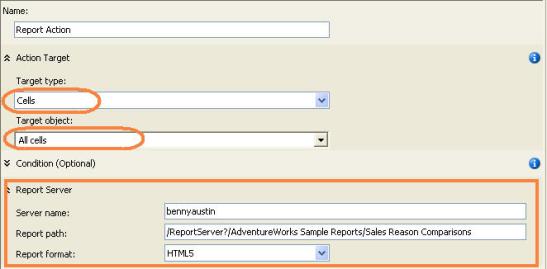


Figure 5 - Target Type and Report URL

4. Now, it’s time to specify the conditions that make the Report visible to the Information Consumer. The condition expression should be a valid MDX expression. In this example, the Report Action should be visible only when the Fiscal Year and Product Category is selected. The MDX expression to achieve that is shown in Figure 6.

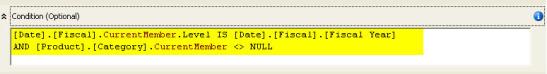


Figure 6 - MDX Condition Expression

|  |  |
| --- | --- |
| 1  2 | [Date].[Fiscal].CurrentMember.LevelIS[Date].[Fiscal].[Fiscal Year] AND[Product].[Category].CurrentMember<> NULL |

5. Next, supply the parameters to the report. As with conditional expression, the parameters should be a valid MDX expression. The parameter name should be same as the one defined in the report (Figure 7). Since the parameter value in the report uses UniqueName, I’m using UniqueName as well.

|  |
| --- |
| SSAS PERSPECTIVES |

# **SSAS Perspectives**

**A perspective is a subset of a cube created for a particular application or group of users. The cube itself is the default perspective. A perspective is exposed to a client as a cube. When a user views a perspective, it appears like another cube. Any changes made to cube data through write back in the perspective are to the original cube.**

Perspectives are a way to reduce the complexity of cubes by hidden elements like measure groups, measures, dimensions, hierarchies etc.  
Perspectives do not require any storage beyond their definition.

Perspectives vs. ViewsPerspectives are similar to views in a relational database in that they narrow down what can be seen from a single cube  
***Perspectives are different from views in that:***They cannot combine multiple cubes to look like one  
They are not securable objects

Working with PerspectivesIn BIDS, if we open an SSAS project there is a Perspectives tab.

This tab shows the cube objects, object type, measure groups, Dimensions and their hierarchies and attributes, KPIs actions and calculations etc

All these items are available for you to turn off and on in a Perspective.

In the Perspective column give a name to the Perspective and tick or untick the items as you want based on the specifications you decide for the end user to see.  
You can create more than one perspective for a SSAS project.

When this is published, the client tools will show both the cube as well as the perspectives.

If excel is used to access the cube, then in excel it shows both the cube and the perspective.

The idea behind this is to simplify their view so that it is simple for them to use.

Perspectives are often most useful when used in pre-formatted reports such as Excel or Performance Point Services analytic Chart and Analytic grids.

# Creating A SSAS 2008-Perspectives

When creating a cube in SSAS with the fact and dimension tables, it is possible that the Measures in the cube could belong to different parts of business.

To elaborate on this, it is possible that the Cube could have Sales related measures, Customer related measures and Product related measures. When the cube is deployed and is being viewed by business users, it is possible that certain business users would like to focus on the measures to their business area.

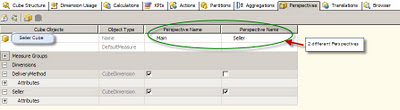
In order to achieve instead of having cubes split into smaller cubes based on business function, there is a feature called Perspectives available in SSAS 2008. Once the cube has been created and related dimensions have been added, the different perspectives can be created. In the screen shot below, the Perspectives tab is highlighted (this is in BIDS/Visual Studio 2008).

[http://3.bp.blogspot.com/-RP5S62yubFE/To8Qnq6HlPI/AAAAAAAACvM/5iWGxxZtT6g/s400/ssas-Perspectives.jpg](http://3.bp.blogspot.com/-RP5S62yubFE/To8Qnq6HlPI/AAAAAAAACvM/5iWGxxZtT6g/s1600/ssas-Perspectives.jpg)

In the Perspectives, initially one gets to see the default perspective which is a view of the whole cube with all the measures and all the dimensions. In order to create a new Perspective, one has to click the New Perspective icon nest to the process icon in the toolbar.

[http://2.bp.blogspot.com/-XR9XyMlUz4c/To8Qvwj1IxI/AAAAAAAACvQ/EkEKua2MTLQ/s400/ssas-Perspectives1.jpg](http://2.bp.blogspot.com/-XR9XyMlUz4c/To8Qvwj1IxI/AAAAAAAACvQ/EkEKua2MTLQ/s1600/ssas-Perspectives1.jpg)

First name the new perspective and start working downwards to start choosing/Checking the Measures needed to be shown for this perspective, likewise do the same for the dimensions.

[](http://1.bp.blogspot.com/-_LvWPvypYd4/To8SUajBBZI/AAAAAAAACvU/YvOhi7W8PM0/s1600/ssas-Perspectives2.jpg)

Once this is completed and the cube is processed, the cube data can be viewed in the Browser tab. In the Browser tab one can choose the different perspectives from the Perspectives dropdown and change the data being viewed. Perspective, I think is a neat way to get different views of data in the cube without physically splitting the cube.

SSAS TRANSLATIONS

# Overview of Translation

A translation is a representation of the names of Microsoft SQL Server Analysis Services (SSAS) objects (such as measure groups, measures, dimensions, attributes, hierarchies, KPIs, actions, and calculated members) in a specific language.

Translations provide server support for client applications that can support multiple languages.

With such a client, the client passes the locale identifier (LCID) to the instance of Analysis Services, which uses the LCID to determine which set of translations to use when providing metadata for Analysis Services objects. If an Analysis Services object does not contain a translation for that language or does not contain a translation for a specified object, the default language is used in returning the object metadata back to the client. For example, if a business user in France accesses a cube from a workstation that has a French locale setting, the business user sees the member captions and member property values in French if a French translation exists. However, if a business user in Germany accesses the same cube from a workstation that has a German locale setting, the business user sees the captions names and member property values in German.

You define translations for dimension metadata on the **Translations** tab in Dimension Designer for the appropriate dimension and on the **Translations** tab in Cube Designer.

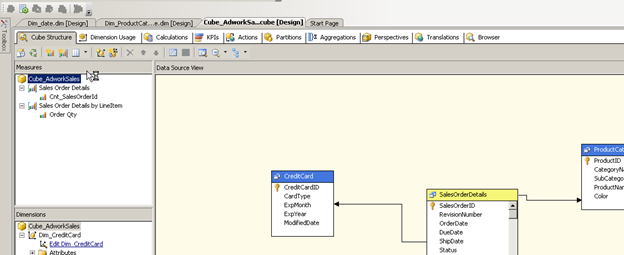
You can use the **Translations** tab of Dimension Designer to define translations not just for the caption of an attribute, but also for the captions of members that are represented by that attribute.

You define translations for dimension metadata on the **Translations** tab in Dimension Designer for the appropriate dimension and on the **Translations** tab in Cube Designer. You can use the **Translations** tab of Dimension Designer to define translations not just for the caption of an attribute, but also for the captions of members that are represented by that attribute.

# Creating a Translation in SSAS

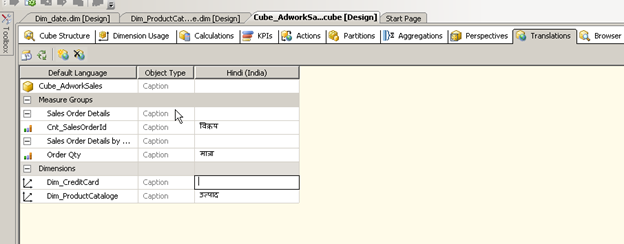
Now let us look at how to augment the cube to provide some additional useful information. For example, let’s assume our cube belongs to a multinational company and as result we need to be able to provide the cube in a local language format that is easily understood by the business users in that geography. While it may be possible that the end business users can manage the do the tasks with some basic working knowledge of the English language the expectation is to provide the cube in a format that is easily understood by the end user and therefore less likely to be misinterpreted.

In order to achieve this we have the ability to add translations within the cube in SSAS. Open up the project for the cube and navigate to the cube designer pane.

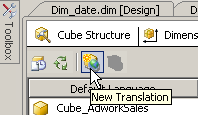


Here we have the tab called Translations on the top.

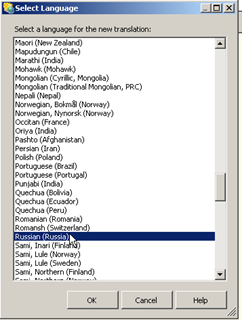
Select the tab to enter the screen below



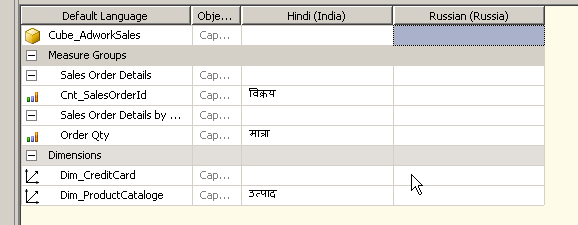
As you can see I have already added some translation for the language Hindi for my dimensions and measures. Let us now add a new set of translations for say Russian.



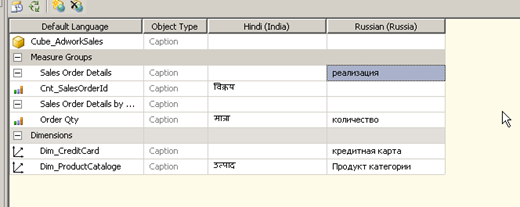
Press the new Translation button the top menu of the window as shown above.



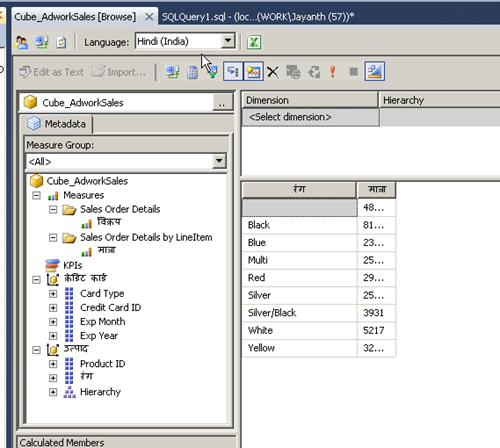
A windows with the list of supported translations appear. Select Russian from the list and press OK.



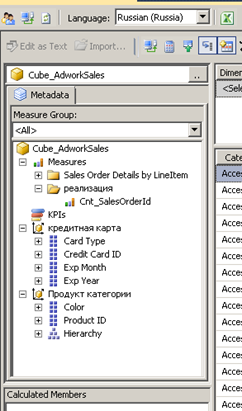
A new column appears right next to the Hindi translations as we are now ready to enter the Russian descriptions for the Different captions we have already specified in English. In this case I will use google translate to identify the Russian equivalent of Quantity (количество)



Now that we have populated the translations in Russian let’s save and deploy the cube. When we browse the cube in SSMS we see there is a Language drop down list on the top of the browser window.

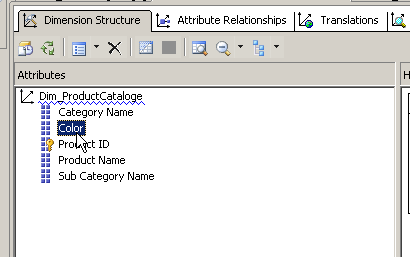


Notice how the language of the captions are in Hindi when I selected the value Hindi in the above drop down list, now change it to Russian.



Here we see that the dimension caption is in Russian but what if we want to Dimension attributes to also display in the local language.

Open up the dimension in the cube designer

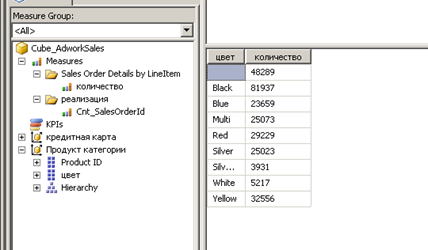


And navigate to the translations tab on the top.

As before select the new translation button on the top and enter the text for the dimension attribute as required.



Save, deploy and process the cube.

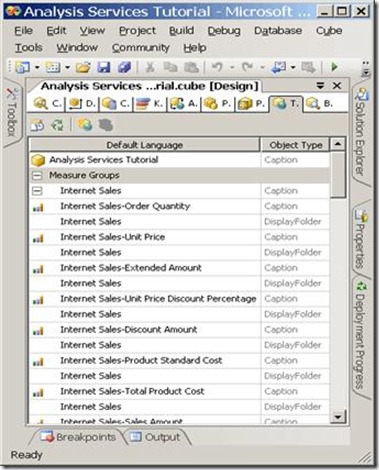


And that’s all there is to it.

Switch to Cube Designer for the Analysis Services Tutorial cube, and then switch to the **Translations** tab.

The metadata in the default language for each cube object appears, as shown in the following image.

The default language in the Analysis Services Tutorial cube is English.



2.On the toolbar of the **Translations** tab, click **New Translation**.

A list of languages appears in the **Select Language** dialog box.

3.Select**Spanish (Spain)**, and then click **OK**.

A new column appears in which you will define the Spanish translations for the metadata objects you want to translate. In this tutorial, we will only translate a small number of objects just to illustrate the process.

4.On the toolbar of the **Translations** tab, click **New Translation**, select **French (France)** in the **Select Language** dialog box, and then click **OK**.

Another language column appears in which you will define French translations.

SSAS PARTITIONS

# Partition

As your database grows in size, Analysis Services cubes that use that database grow along with it. As such, one thing that can improve performance of your cube is partitioning (splitting up) your measures.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1990-95 | 1995-2000 | 2000-2005 | 2010-2015 | 2015 -> |  |  |  |
| Query |  |  |  |  |  |  |  |

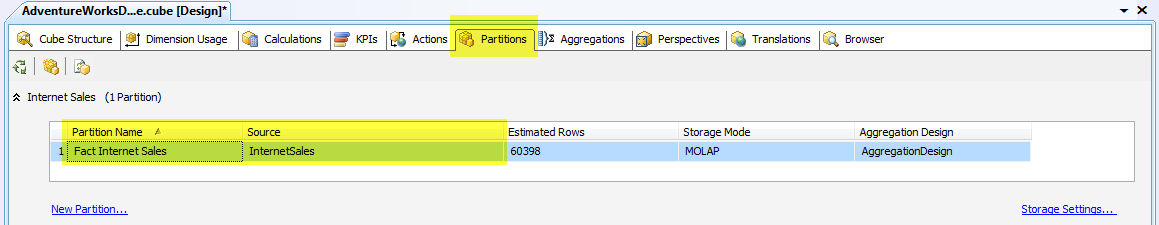
**Partition Benefits:**

* **Better Query Performance**: Cube partition is a powerful mechanism for improving query performance. Queries that summarize data over 10 years could take considerably longer than those that only search through the current year data. If we have proper partitions then SSAS only has to scan a small subset of data to return query results hence dramatic performance improvements compared to queries running against a cube with a single partition.
* **Minimize downtime**:  Cube partitioning supports reducing downtime associated with cube processing. In almost all the cases, a portion of data warehouse is volatile and needs to be processed often. However other portions are relatively static. For example, in a sales cube, we need to change the current year's data nightly, but sales from previous years might change only occasionally - in case if account for merchandise returns and exchanges. If your warehouse tracks last 10 years salesthen processing only the current partition may be 10 times quicker than processing the entire cube.
* **Aggregations benefits**: The partition queried frequently could benefit from additional aggregations, which in turn could improve performance. Partition(s) that are used less can be processed less frequently with considerably fewer aggregations.
* **Customized storage and processing settings**: Frequently accessed partitions might benefit from proactive caching and ROLAP storage. On the other hand, other forms of storage and processing might be better for less frequently queried partitions.
* **Distributed query and processing load**: SSAS allows you to create remote partitions - a remote partition resides on a server different from its parent cube. This way the queries that affect the remote partition are processed on a server separate from its parent cube, allowing you to take advantage of additional processing power.
* **Parallel Partitions Processing**: SSAS allows processing multiple partitions in parallel on a server that has multiple processors. This can further reduce the total cube processing time.

# Creating a Partition in SSAS

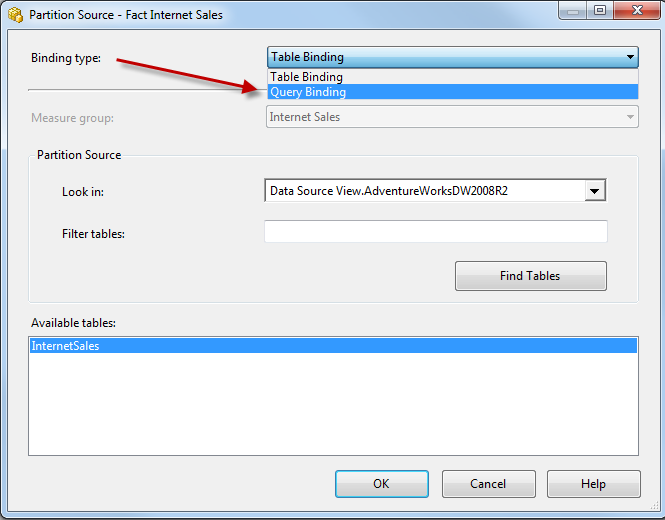
When you have your cube opened, navigate to the Partitions tab. A screenshot of what this tab looks like is shown in figure 1. You can have a partition set up for the entire table (not split up) or you can write queries that would return the data you want to be included in that partition.

**Figure 1: Partitions Tab**



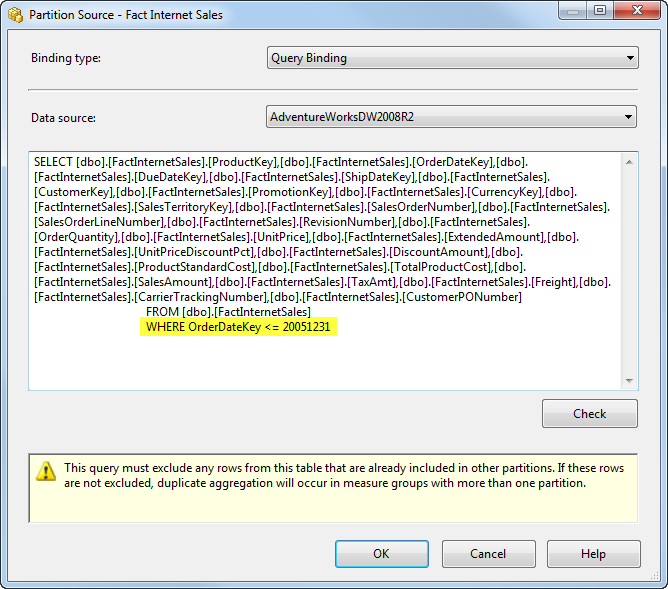
For my example, I am going to break out my Internet Sales measure by years. Which for the AdventureWorksDW2008R2.dbo.FactInternetSales table, we have 2005 – 2008. So, I’m going to create four partitions starting with 2005. To switch the default table binding partition to query mode, select “Query Binding” binding type as shown in figure 2 below.

**Figure 2: Switching the Binding Type**



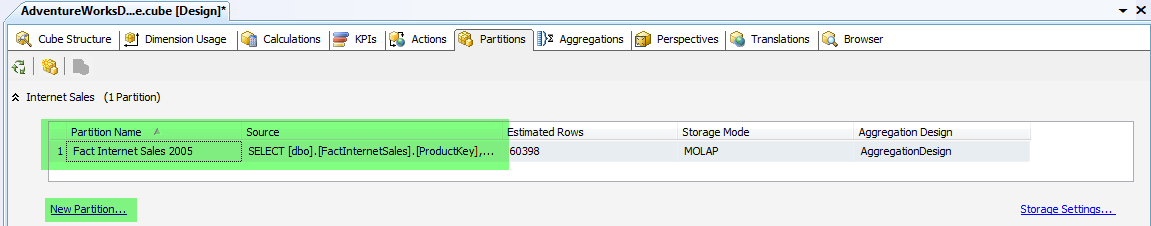
After you switch it to query binding, you’ll be shown a query that you want this partition to contain. Since the first set of Internet Sales records were in 2005, I am just going to update the script to contain “WHERE OrderDateKey<= 20051231” as shown in figure 3.

**Figure 3: Update the Script**



After modifying the name assigned to my new partition to “Fact Internet Sales 2005” and choosing to design aggregations later, I now have the results shown in figure 4.

**Figure 4: Partition Created**



Now, I need to click on the “New Partition…” link to create my other partitions in a similar fashion. The slimmed down scripts used to create each of these partitions are shown in script 1.

**Script 1: Queries to Partition by Year**

--2005 Internet Sales Partition Query

SELECT \*

FROM [dbo].[FactInternetSales]

WHERE OrderDateKey <='20051231'

--2006 Internet Sales Partition Query

SELECT\*

FROM [dbo].[FactInternetSales]

WHERE OrderDateKey >='20060101'AND OrderDateKey <='20061231'

--2007 Internet Sales Partition Query

SELECT\*

FROM [dbo].[FactInternetSales]

WHERE OrderDateKey >='20070101'AND OrderDateKey <='20071231'

--2008 Internet Sales Partition Query

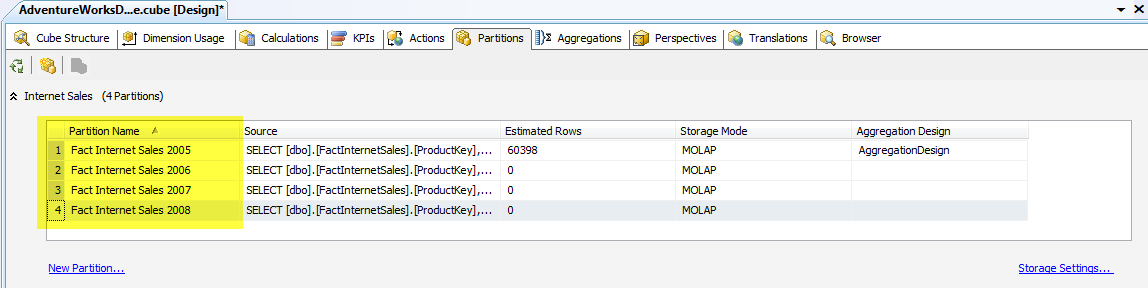
SELECT\*

FROM [dbo].[FactInternetSales]

WHERE OrderDateKey >='20080101'AND OrderDateKey <='20081231'

The results after creating all of these partitions should look like that shown in figure 5 below.

**Figure 5: All Yearly Partitions Created**



That’s all there is to creating partitions. So, after assigning these partitions to my Aggregation Design as shown in the example above, deploying and reprocessing, accessing the data within the partitions should be much quicker than having to search through a massive table containing hundreds of millions of records.

# WRITEBACKS IN CUBES

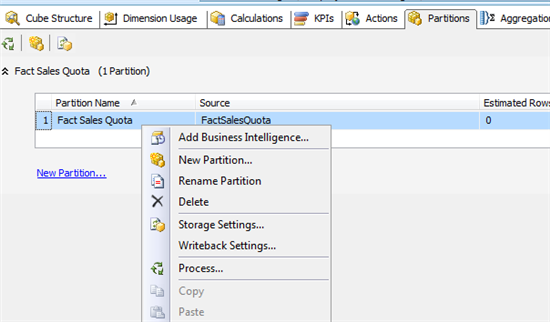
Writeback enhancement to a cube or dimension to allow users to manually modify the dimension structure and members. Updates to a write-enabled dimension are recorded directly in the dimension table. This enhancement changes the WriteEnabled property setting for a dimension.

To add dimension writeback, you use the Business Intelligence Wizard, and select the Enable dimension writeback option on the Choose Enhancement page. This wizard then guides you through the steps of selecting a dimension to which you want to apply dimension writeback and setting this option for the selected dimension.

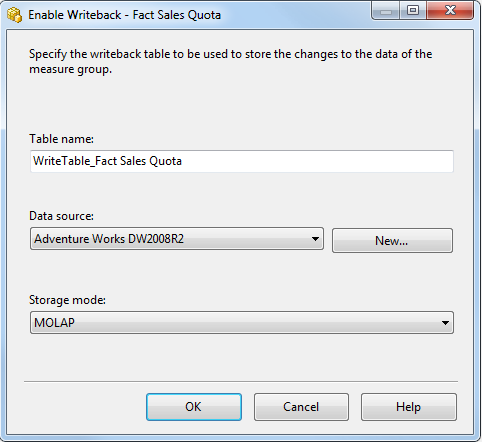
Note : Write backs is only possible in star schema.In snow flake schema writebacks. Excel is one of the tool through which we can make changes to the cube data through writebacks.

CREATING WRITEBACKS IN CUBE:

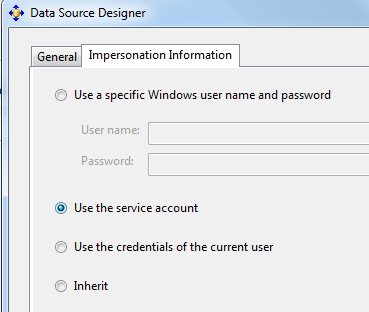
To enablewrite back in a cube. Go to the partion tab and right click the existing partion. Now you can specify the writeback settings



Specify the table name for the writeback table

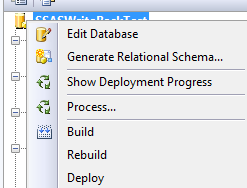


The next step necessary would be to set the impersonation account settings:

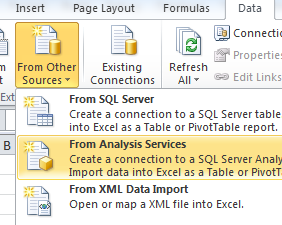


You need to provide an account that has the rights to create a table in the data source. In my case I use the service account.

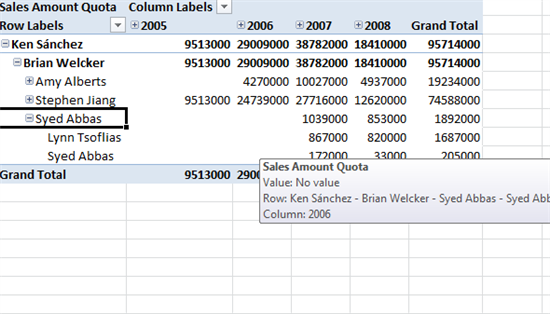
Now the cube can be deployed and processed



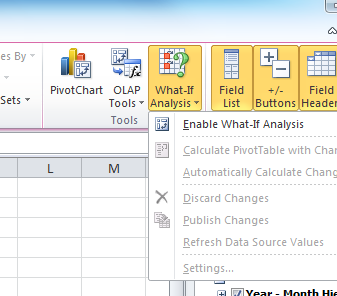
 Now you can start Excel 2010 and create a connection to the cube:



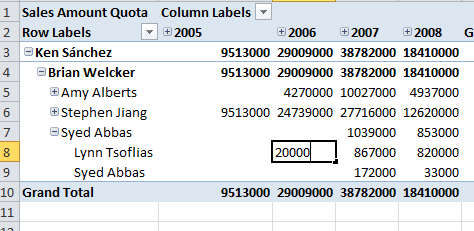
 Create a PivotTable with the following characteristics



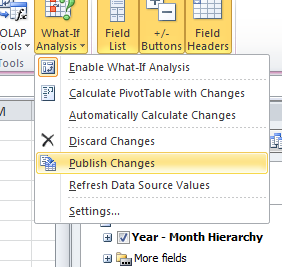
 In order to enable write back to the cube it has to be enabled on the PivotTable, this can be done on the What-if Analysis button on the PivotTable Options tab.



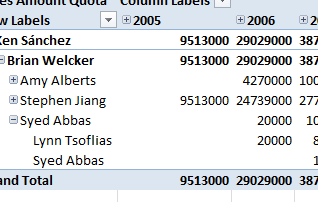
 After this has been done you can write to the cells in the PivotTable



 This will change the PivotTable but not write the data to the database. In order to write the changes to the database you need to publish your changes:



 Now the data has been written to the writeback table including the time it was changed and by whom



 Now you go back to the cube and see the changes reflected in the cube.

# PROACTIVE CACHING

Mostly data warehousing and Business Intelligence (BI) applications work on historical data, which are normally refreshed on a daily, weekly or monthly basis. Although there are some near real time data warehousing and BI applications, their query performance is painfully slow. To help in this scenario, Analysis Services (Microsoft SQL Server 2005 and later versions) provides an amazing feature called "Proactive Caching". This new feature provides a query performance similar to that of historical data, with near real time data and minimal impact on the overall system. As a BI developer, you just need to configure proactive caching, and SQL Server Analysis Services (SSAS) will ensure you get performance of a fully processed data warehouse on near real time data.

## How it works

When you configure proactive caching for a cube, a measure group partition or a dimension, SSAS starts listening to data change notifications from the underlying relational database.On the relational database side, if there is any change in the data of the related objects, an event is generated and notification is sent. SSAS intercepts this notification and start refreshing the MOLAP cache. Depending on your configuration, during that time frame SSAS either drops the Multidimensional OLAP structure (MOLAP cache) and serves the query request from the underlying relational database (behaves as ROLAP storage mode) or creates another MOLAP cache in parallel to the outdated one.

As long as the new MOLAP cache is being built, the queries are served from the outdated MOLAP cache (of course query result will be based on old data while the new MOLAP cache is being built). The moment the new refreshed MOLAP cache is ready, all queries are switched back to this new MOLAP cache and the outdated MOLAP cache is dropped. This process of synchronization happens in the background so there is minimal impact on the overall system.

While configuring the proactive caching you can specify the following:

* How frequently the MOLAP cache is to be rebuilt
* From where SSAS should get data while the data refresh is in progress; it should get it from underlying relational database or from the outdated MOLAP cache.
* The MOLAP cache should be refreshed automatically when a notification arrives or on specified interval/scheduled basis.

After enabling proactive caching, you get the best query performance of MOLAP storage mode on the most recent data.

You should consider if your clients want to see recent data without compromising with the query performance.

## MOLAP

In MOLAP storage mode (default mode), detail data and aggregations are stored in multidimensional format at the SSAS server and no notifications are received by the SSAS server because proactive caching does not work in this mode. MOLAP storage mode is applicable if you want the best query performance, freshness of data is not that important and you have enough storage space on the SSAS server. You can either schedule processing of MOLAP cache or process it manually whenever required. Apart from this mode, all other storage modes discussed below support enabling proactive caching.

## Scheduled MOLAP

Like MOLAP storage mode, in Scheduled MOLAP mode the detail data and aggregations will be stored in multidimensional format at the SSAS server. The only difference is, MOLAP cache is automatically rebuilt on a 24 hour schedule by SSAS itself (SSAS does not listen for change notification from relational data source; it just does the processing every 24 hours). As a Data Warehouse/BI developer, to do the processing you aren't required to create an SSIS Package or SQL Server agent job with this mode.

Scheduled MOLAP is suitable for applications that have daily data refresh requirement.

## Automatic MOLAP

Again, like MOLAP storage mode, in Automatic MOLAP mode the detail data and aggregations will be stored in multidimensional format at the SSAS server but in this mode, SSAS listens for change notifications from the underlying relational database source. If any update happens in the relational database, an event is triggered and SSAS is notified about the data changes. On receiving this change notification, SSAS automatically starts refreshing the cube or partition data. It actually a creates a new MOLAP cache and also maintains the old one in parallel so that the existing request can be served from the outdated/old MOLAP cache. The moment creation and processing of new MOLAP cache is complete, all query requests are switched back to the new MOLAP cache and the old one is dropped.

If you want processing to occur, to refresh the data in MOLAP, with a latency interval of 2 hours and if you also want high query performance, this mode would be your choice. During rebuild and processing of the new MOLAP cache, the query gets stale data as it is accessing the outdated MOLAP cache.

With this mode there are two important settings (Silence Interval and Silence Override Interval) that you need to be aware of. Silence Interval has a default value of 10 seconds, which means after receiving a change notification SSAS waits for another 10 seconds to see if there is any more change notification coming (this ensures more updates can be considered in a single processing). If another change notification does not arrive in 10 seconds, SSAS will start processing the cube or partition. If any change notification arrive within 10 seconds then it waits for another 10 seconds until the Silence Override Interval is reached (default is 10 minutes). Silence Override Interval is required because without Silence Override Interval, your cube will never be processed if continuous updates are happening at relational data source.

## Medium Latency MOLAP

Like the MOLAP storage mode, in this mode the detail data and aggregations will be stored in multidimensional format at the SSAS server. Also, as in case of Automatic MOLAP, SSAS listens for change notifications from the underlying relational database.

On receiving this change notification, SSAS automatically invalidates the existing MOLAP caches and starts reprocessing it (notice it does not create another MOLAP cache in parallel to the old one, as in the case of Automatic MOLAP). While reprocessing, the queries are switched to real time ROLAP (discussed in the next article) and once the processing of the MOLAP cache is complete; queries are again switched back to the MOLAP cache. Since during reprocessing real time ROLAP is used to serve the queries, performance would obviously be slower during this time frame.

This mode has a target latency of 4 hours; it means SSAS will drop out the MOLAP cache after 4 hours of receiving any change notification from the underlying relational database. Similar to automatic MOLAP, in this case Silence Interval has a default value of 10 seconds and Silence Override Interval has a default value of 10 minutes.

You would use this storage mode when the relational data source is updated infrequently and query performance is important for your users.

## Low Latency MOLAP

As you might have guessed, Low Latency MOLAP is, to some extent, similar to Medium Latency MOLAP mode. The only basic difference is the latency; in this mode, the processing occurs automatically as data changes with a target latency of 30 minutes after the first change notification is received.

A point to note here is that since this mode does switching to real time ROLAP frequently (30 minutes after a change notification), the performance of a query would be poorer.

You would use this storage mode when the relational data source is updated frequently and refreshed data is more important than query performance.

# Real-time HOLAP

Real-time HOLAP mode is similar to the standard HOLAP storage mode in terms of the data storage location; Detail data is stored in the relational database in relation format whereas the calculated aggregations are stored in multidimensional format on the OLAP server. The difference between this mode and standard HOLAP storage mode is that proactive caching is enabled in this mode. In other words, in this mode the SSAS server listens for any data change notification coming from the underlying relational database. When this notification is received by SSAS, queries are switched to real time ROLAP and MOLAP aggregations are refreshed. Once this MOLAP aggregation refreshment is finished, the queries are again switched back to Real-time HOLAP. This mode has a Silence Interval of 0 seconds and no Silence Override Interval and hence it has zero latency.

**Real-time ROLAP**

Coming to the last storage mode available in proactive caching, we have Real-time ROLAP. Again, this mode is similar to standard ROLAP storage mode in terms of storage location. **Both the detail data and aggregations are stored in the relational format in the underlying relational database.** The difference is, in this mode, SSAS listens for change notification from the underlying relational database. When this notification is received by SSAS, it uses the changed data and, as a result of this, all query results are based on current data (zero latency).

Use this storage mode when the relational data source is updated frequently and the latest data is required by the users. In this mode, query performance is slower than all discussed modes as SSAS retrieves all of the detail data and aggregations from the underlying relational database.

Advantage : will always get the latest data, because the data will be coming from the relational database.

Disadvantage : the query performance will be slow.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Storage Mode** | **Enable Proactive Caching** | **Silence Interval** | **Silence Override Interval** | **Latency** | **Rebuild Interval** | **Bring Online Immediately** |
| **MOLAP** | MOLAP | No |  |  |  |  |  |
| **Scheduled MOLAP** | MOLAP | Yes |  |  |  | 1 Day |  |
| **Automatic MOLAP** | MOLAP | Yes | 10 Sec | 10 Min |  |  |  |
| **Medium Latency MOLAP** | MOLAP | Yes | 10 Sec | 10 Min | 4 hours |  | Selected |
| **Low Latency MOLAP** | MOLAP | Yes | 10 Sec | 10 Min | 30 Min |  | Selected |
| **Real Time HOLAP** | HOLAP | Yes | 0 Sec |  | 0 Sec |  | Selected |
| **Real Time ROLAP** | ROLAP | Yes | 0 Sec |  |  |  | Selected |
| Comment : | | | | | | | |

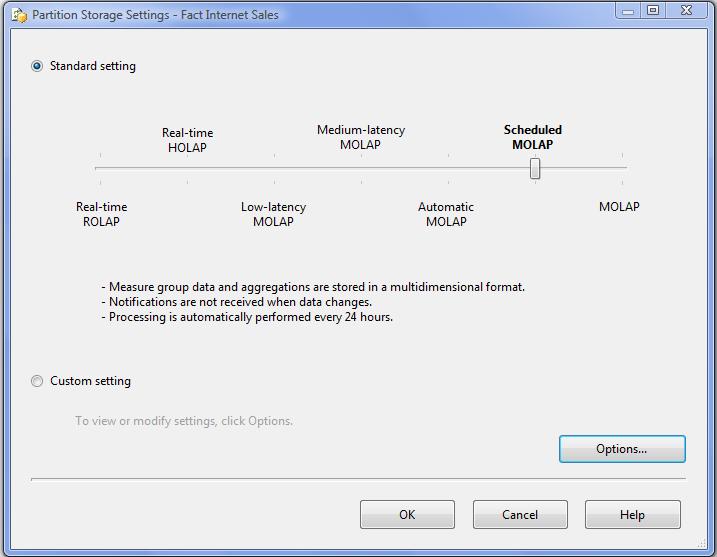
***Note: Storage Modes needs to be set separately for Dimensions and Cube.***

# Configuring proactive caching

The proactive caching feature ensures a cube or a measure group partition or dimension reflects current data on its own. Depending on your choice of proactive caching enabled storage mode, interception of data change notification, storage location and update frequency will vary.

You use the **Storage Settings** dialog box in BIDS (Business Intelligence Development Studio) to set the proactive caching feature, storage location, and notification settings for a dimension, cube, measure group, or measure group partition. Though there are several paths to reach the **Storage Settings** dialog box, I am providing one of them below, to enable proactive caching for a partition.

1. In the Cube Browser, open your cube and select the Partitions tab.
2. Expand the measure group and select the partition for which you want to enable proactive caching.
3. Click the Storage Setting link to open the Partition Storage Settings dialog box similar to the one as shown in below image.

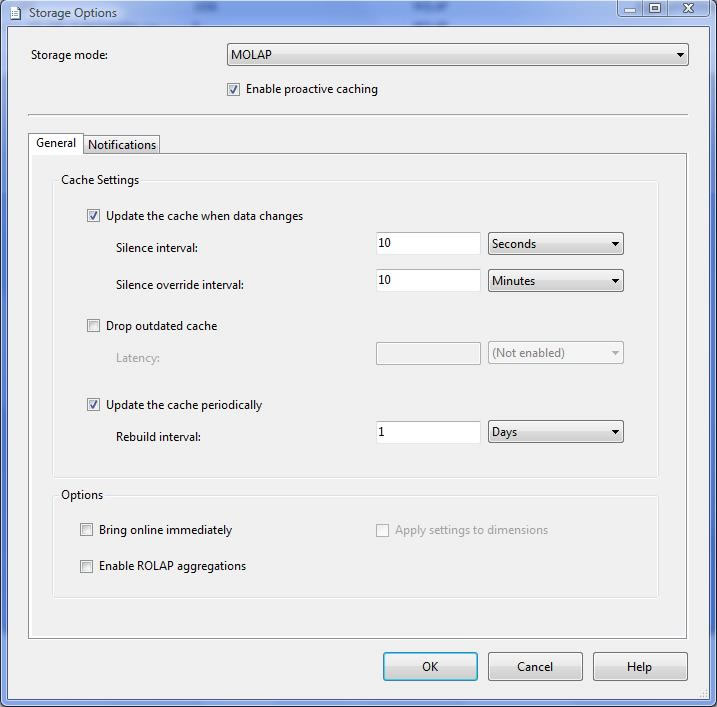
  
**Figure 1 - Enabling Proactive Caching**

1. Move the Standard Setting slider to storage mode as per your requirement.
2. The Custom Setting allows you to explicitly enable proactive caching (if you don't want to use Standard Setting), set storage mode, and notification options.

When you use proactive caching, there are several settings, as discussed below, which you can manually tune, like the target latency, the notification listening intervals and the notification types.

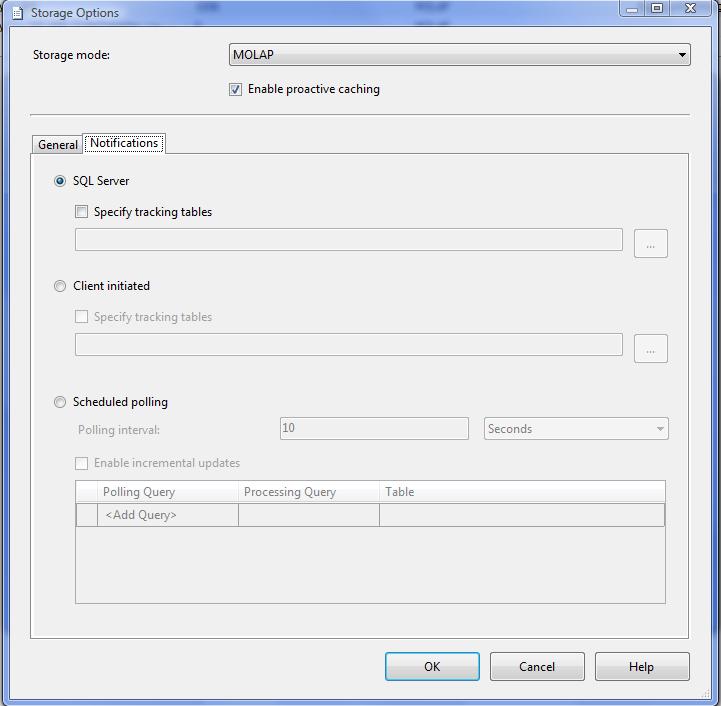
Please note some of the settings will not be available on some of the objects.

* *Silence Interval* - SSAS waits for a set amount of time after receiving a change notification to see if there are more change notifications coming. With this setting, SSAS ensures as many as possible can be incorporated in a single refresh cycle.
* *Silence Override Interval* - Because of Silence Interval, if there are continuous changes happening, your cube will never be processed. To override this we have the Silence Override Interval setting.
* *Latency* - Latency is the maximum amount of time SSAS waits to create a new MOLAP cache. Once latency has been reached, SSAS drops the existing MOLAP caches and starts making another one; during this time frame (during processing) queries are routed to relational data source.
* *Update the cache periodically* - This is the additional setting, which if selected, SSAS will ensure the MOLAP cube is refreshed periodically irrespective of any data change notification.
* *Bring online immediately* - If you select this option, queries are routed to the underlying relational data source while the MOLAP cache is being rebuilt.
* *Enable ROLAP aggregations* - If you select this option, indexed views are created at the relational database for aggregations.
* *Apply settings to dimensions* - If you select this option, the same proactive caching setting will be applied on related dimensions as well.

  
**Figure 2 - Proactive Caching - General settings**

On the Notifications tab, there are three options out of which, as shown below, you can select any one at a time.

* SQL Server - With this option, SSAS uses SQL Server notification services/specialized trace mechanism to identify data changes.
* Client initiated - With this option, client can specify the XMLA (XML for Analysis) command (NotifyTableChange) to identify data changes.
* Scheduled polling - With this option, SSAS uses a series of queries to see (polling at defined interval) if there is any data change at the underlying relational database.

  
**Figure 3 - Proactive Caching - Notification settings**

Proactive caching provides developer automatic management of cube processing when changes happens on relational database. Proactive caching feature provides different standard settings on how frequently you want to detect changes and get real time data.

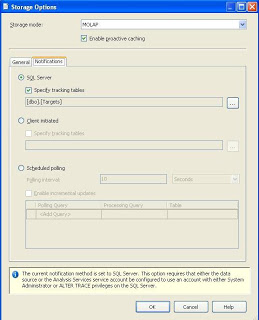
Example

Let me take example of "Automatic  MOLAP" where different parameter values are:

1) Silence Interval = 10 seconds  
2) Silence Override Interval = 10 minutes

|  |
| --- |
| http://1.bp.blogspot.com/_ntCExJSVw8o/TLVudwWNdMI/AAAAAAAAMX8/HLvSvyrTKwo/s320/PC_AutomaticMOLAP1.JPG |
| **Snapshot1** |

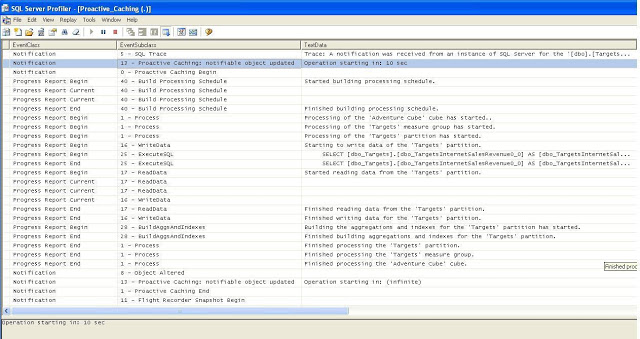
I have set "SQL Server Notification" on "Targets" table.



Now deploy changes on cube and we are ready for receiving changes from relational database. Lets configure "SQL Server Profiler" on analysis server so we can track all the changes happening on the server.

We will change Target table data and see events in profiler:

UPDATE [dbo].[TARGETS]  
SET InternetSalesRevenue = 15000000  
WHERE CalendarYear = 2005



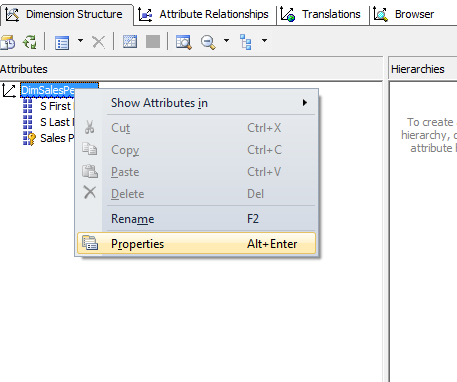
**Step 1:** Analysis Services recieves notification on changes in "Target" table.

**Step2:** Analysis Services waits for "Silence Interval" time period.

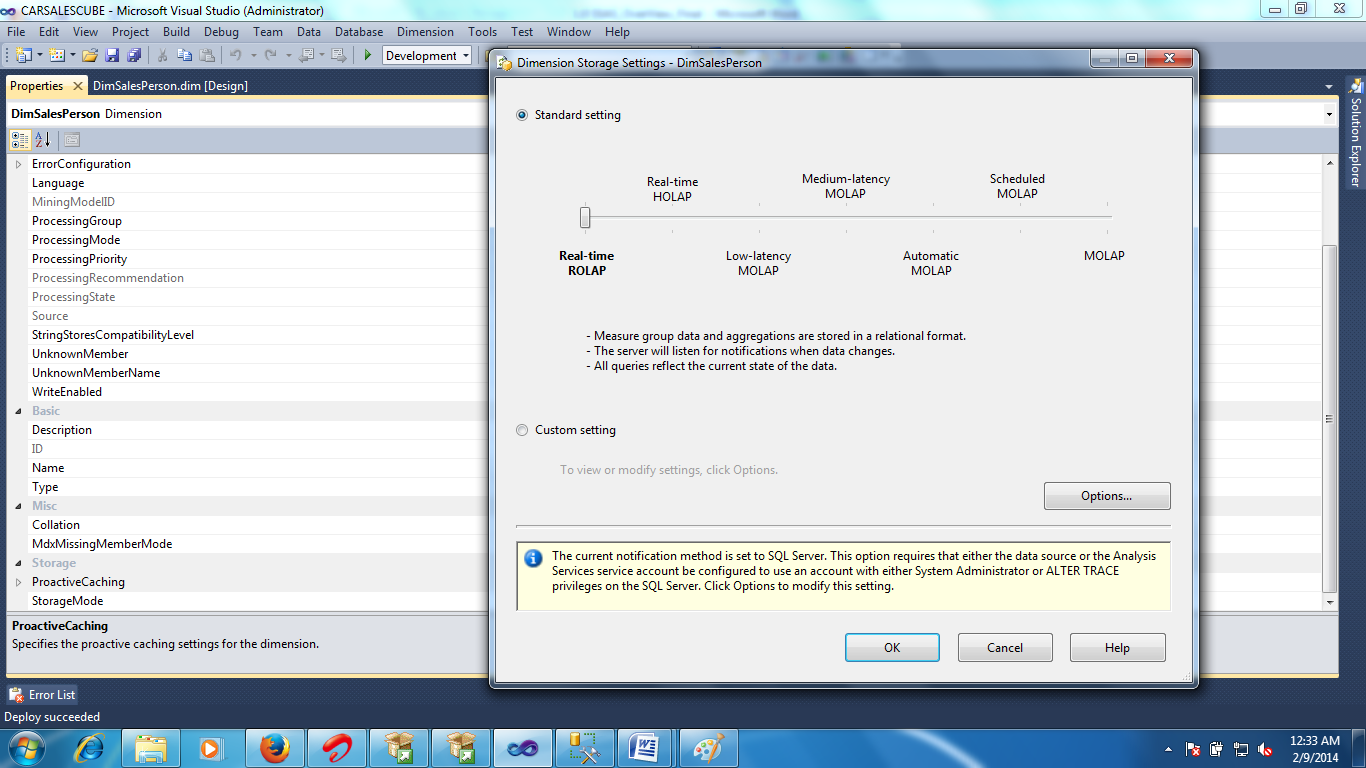
**Step3:** If no changes detected in "Silence interval" time period then it reprocess the cube.

# CHANGING STORAGE MODE FOR DIMENSIONS

* 1. Selection the Dimension and go to the Properties :



* 1. In the Properties section go to Proactive Caching section and select the Storage Mode.



# Aggregations

Aggregations are precalculated summaries of data from leaf cells.

Aggregations improve query response time by preparing the answers before the questions are asked.

For example, when a data warehouse fact table contains hundreds of thousands of rows, a query requesting the weekly sales totals for a particular product line can take a long time to answer if all the rows in the fact table have to be scanned and summed at query time to compute the answer. However, the response can be almost immediate if the summarization data to answer this query has been precalculated. This precalculation of summary data occurs during processing and is the foundation for the rapid response times of OLAP technology.

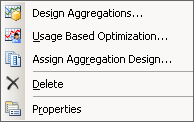
Cubes are the way that OLAP technology organizes summary data into multidimensional structures. Dimensions and their hierarchies of attributes reflect the queries that can be asked of the cube. Aggregations are stored in the multidimensional structure in cells at coordinates specified by the dimensions. For example, the question "What were the sales of product X in 1998 for the Northwest region?" involves three dimensions (Product, Time, and Geography) and one measure (Sales). The value of the cell in the cube at the specified coordinates (product X, 1998, Northwest) is the answer, a single numeric value.

Other questions may return multiple values. For example, "How much were the sales of hardware products by quarter by region for 1998?" Such queries return sets of cells from the coordinates that satisfy the specified conditions. The number of cells returned by the query depends on the number of items in the Hardware level of the Product dimension, the four quarters in 1998, and the number of regions in the Geography dimension. If all summary data has been precalculated into aggregations, the response time of the query will depend only on the time that is required to extract the specified cells. No calculation or reading of data from the fact table is required.

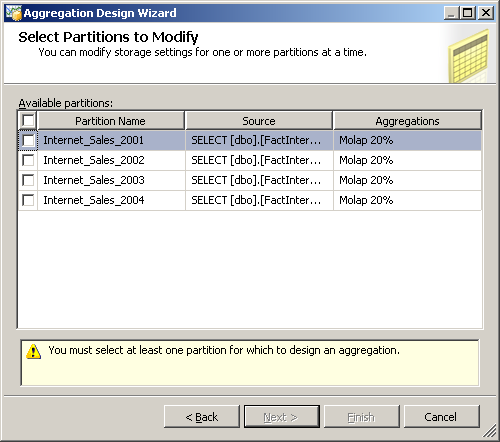
Although precalculation of all possible aggregations in a cube might provide the fastest possible response time for all queries, Analysis Services can easily calculate some aggregrated values from other precalculated aggregations. Additionally, calculating all possible aggregations requires significant processing time and storage. Therefore, there is a tradeoff between storage requirements and the percentage of possible aggregations that are precalculated. If no aggregations are precalculated (0%), the amount of required processing time and storage space for a cube is minimized, but query response time may be slow because the data required to answer each query must be retrieved from the leaf cells and then aggregated at query time to answer each query. For example, returning a single number that answers the question asked earlier ("What were the sales of product X in 1998 for the Northwest region") might require reading thousands of rows of data, extracting the value of the column used to provide the Sales measure from each row, and then calculating the sum.

Right-clicking on an aggregation design gives a number of options including

"Design New Aggregations", "Usage Based Optimisation" and "Assign Aggregation designs".

[](http://geekswithblogs.net/images/geekswithblogs_net/darrengosbell/WindowsLiveWriter/SSAS2008NewCubeAggregationstab_9F7A/image2.png)

When you start up the "Design Aggregations" Wizard you can now easily select which aggregations a particular design is to apply to.



One of the coolest additions to the aggregation design wizard is the following screen which shows you the aggregation usage setting for all the attributes and gives you the ability to change them.



Review Aggregation Usage:

In this screen it will show all the dimensions taking participation while creating aggregation with options to be select against each attribute of each dimension.

* *Default:* Internally it will apply the default value based on the type of attribute and dimension.
* *Full:* In every aggregation of the cube the full option selected attribute will be included. This must be avoided because if the attribute has multiple members then it may prevent from the aggregation due to increase in size.
* *None:* This attribute will not take part in the aggregation.
* *Unrestricted:* The attribute will get the unrestricted during the aggregation but still it should be evaluated.

The *AggregationUsage* property controls how dimension attributes are treated in the aggregation design process. The property can be set to the following values:

* **Full**: This means the attribute, or an attribute at a lower granularity directly related to it by an attribute relationship, will be included in every single aggregation the wizard builds. We recommend that you use this value sparingly, for at most one or two attributes in your cube, because it can significantly reduce the number of aggregations that get built. You should set it for attributes that will almost always get used in queries. For example, if the vast majority of your queries are at the Month granularity it makes sense that all of your aggregations include the Month attribute from your Time dimension.
* **None**: This means the attribute will not be included in any aggregation that the wizard designs. Don't be afraid of using this value for any attributes that you don't think will be used often in your queries, it can be a useful way of ensuring that the attributes that are used often get good aggregation coverage. Note that Attributes with *AttributeHierarchyEnabled* set to *False* will have no aggregations designed for them anyway.
* **Unrestricted**: This means that the attribute may be included in the aggregations designed, depending on whether the algorithm used by the wizard considers it to be useful or not.
* **Default**: The default option applies a complex set of rules, which are:
  + The granularity attribute (usually the key attribute, unless you specified otherwise in the dimension usage tab) is treated as Unrestricted.
  + All attributes on dimensions involved in many-to-many relationships, unmaterialised referenced relationships, and data mining dimensions are treated as None. Aggregations may still be built at the root granularity, that is, the intersection of every All Member on every attribute.
  + All attributes that are used as levels in natural user hierarchies are treated as Unrestricted.
  + Attributes with *IsAggregatable* set to *False* are treated as Full.
  + All other attributes are treated as None

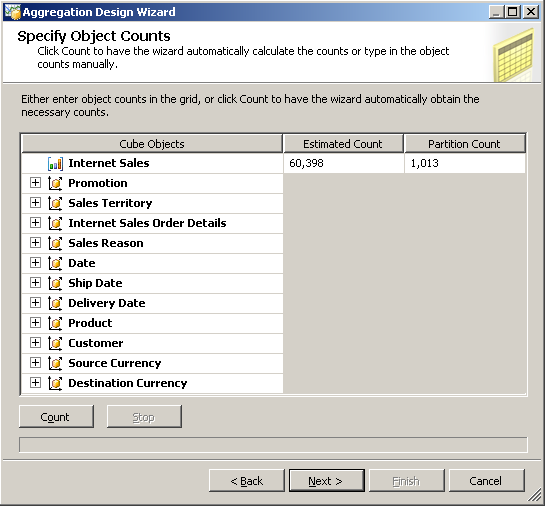
Tips:

* Set the AggregateUsage property to **Unrestricted** only for the frequently-used attribute hierarchies: The Best Practices paper suggests five - ten **Unrestricted** attributes per dimension.
* If the attribute is contained in most of the queries, then set **AggregateUsage** should be set to **Full**. This only applies to attributes where there are only a few members; otherwise, the wizard will take a long time to devise a design, and implement it.
* If the attribute isn't used often, then this should be set to **None**. This reduces the overall search space.

Specify Objects Counts: In this screen it will show all the dimensions taking participation while creating aggregation with count of rows.

* *EstimatedCount*
* *PartitionCount*

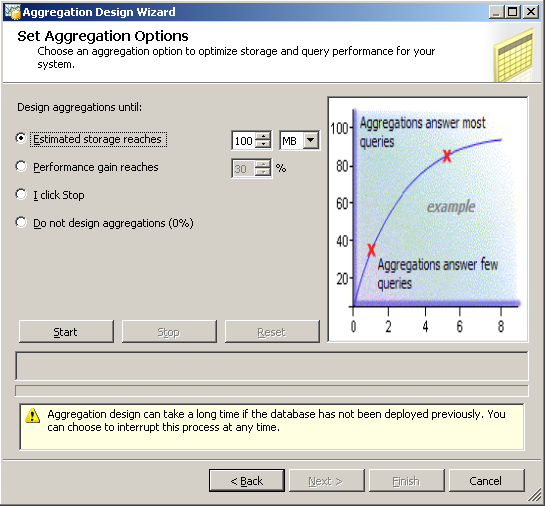
The objects count screen appears largely unchanged.



As does the core Aggregation Options screen.

Set Aggregation Options:  In this screen to select options to optimize storage and query performance gain of the cube.

* *Estimated Storage reaches*
* *Performance gain reaches*
* *I click stop*
* *Do not design Aggregations*



On the Set Aggregation Options step you finally reach the point where some aggregations can be built. Here you can apply one last set of restrictions on the set of aggregations that will be built, choosing to either:

* **Estimated Storage Reaches**, which means you build aggregations to fill a given amount of disk space.
* **Performance Gain Reaches**, the most useful option. It does not mean that all queries will run n% faster; nor does it mean that a query that hits an aggregation directly will run n% faster. Think of it like this: if the wizard built all the aggregations it thought were useful to build (note: this is not the same thing as all of the possible aggregations that could be built on the cube) then, in general, performance would be better. Some queries would not benefit from aggregations, some would be slightly faster, and some would be a lot faster; and some aggregations would be more often used than others. So if you set this property to 100% the wizard would build all the aggregations that it could, and you'd get 100% of the performance gain possible from building aggregations. Setting this property to 30%, the default and recommended value, will build the aggregations that give you 30% of this possible performance gain—not 30% of the possible aggregations, usually a much smaller number. As you can see from the screenshot below, the graph drawn on this step plots the size of the aggregations built versus overall performance gain, and the shape of the curve shows that a few, smaller aggregations usually provide the majority of the performance gain.
* **I Click Stop**, which means carry on building aggregations until you click the Stop button. Designing aggregations can take a very long time, especially on more complex cubes, because there may literally be millions or billions of possible aggregations that could be built. In fact, it's not unheard of for the aggregation design wizard to run for several days before it's stopped!
* **Do Not Design Aggregations** allows you to skip designing aggregations.

At the end of the whole process you get to give your design a name. I would suggest that you might want to think about including an indication of the design options that you used. (eg "30 pcnt", "upto 150Mb", etc. ) You can see a property for the aggregation level when you view the properties of the aggregation design, which is where I thought information like this would be recorded, but that property did not appear to update (maybe this will change in a future build). But if you have a couple of different aggregation designs, this would let you see at a glance, which was which.

