Cube, unified dimensional model (UDF)

--definition

A cute is a logical storage object that combines dimensions and measures to provide a multi-dimension view of data. It usually created on top a data warehouse and provide ad-hop analysis service. 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.

Example: you may slice data by dimensions or you may aggregation data across a hierarchy. A cute often stores pre-aggregated data to enable faster performance

SSAS 2012 new feature

--Tabular Project

SSAS components

--Data Source

It represent a connection to the database where the data is stored. You can build a data source view on top of data sources. To create a date source, you will need to specify a connection string as well as the impersonation type. **It only support native OLE DB and .net provider.** Among which OLE DB should be used whenever possible for performance issues. Following are some of the major sources supported by SSAS: SQL Server, MS Access, Oracle, Teradata, IBM DB2, and other relational databases with the appropriate OLE DB provider.

--Date source views

It is a logical view of the underlying database schema. It abstract the underlying database schema. DSV is very useful by letting you augment the schema. For example, you can add calculated column to DSV when security policies prevent you from changing the database schema.

--Dimensions

--Cubes

--Mining Structures

--Roles

--Assemblies

--Miscellaneous

Impersonation option

* Impersonation allows SSAS to assume the authentication to perform **the server side** data operations like data access, [processing](https://www.mssqltips.com/sqlservertutorial/2011/processing-dimensions-and-cube/) etc. As part of impersonation, the following options are available in SSAS:
  + *Use a specific Windows user name and password*: Thi+\*+
  + s option lets you to specify **Windows account credentials** which will be used by SSAS to perform operations like source data access, processing etc.
  + *Use the service account*: When this option is selected, SSAS uses the credentials of the **service account under which the Analysis Services service is configured/running for source data access, processing etc.**
  + *Use the credentials of the current user*: When this option is set, SSAS uses the credentials of the current user for performing operations like DMX Open Queries, Local cubes etc. This option cannot be used for performing server side operations like source data access, processing etc.
  + *Inherit*: This option let's the SSAS server decide which impersonation mode is suitable for each type of operation. When this option is set, by default SSAS will use the service account for operations like processing and the credentials of the current user for operations like Local cubes, querying the data mining models, etc...

Steps to develop a cube

* Reading data from a dimensional model.(DATA SOURCE)
* Configuring a schema in BIDS (SQL Server Data Tools). (DSV)
* Creating dimensions, measures and cubes from this schema.
* Fine tuning the cube as per the requirements
* Deploying the cube.

Is it possible to combine data from multiple data sources in SSAS? If yes, how do you accomplish it?

SSAS allows combining data from multiple underlying data sources into a single DSV. To be able to add Table(s)/View(s) from multiple data sources, first you need to create a DSV using your first source and this source acts as the primary data source. Now after the initial DSV is created, you can add one or more data sources into DSV which will act as secondary data sources and you can choose additional Table(s)/View(s) from the secondary data sources which you want to include in your DSV.

The key thing while combining data from multiple data sources is that the Primary Data Source must support OPENROWSET queries. Hence in most cases, SQL Server is used as the Primary Data Source.

**Named query and Named calculation**

**--Named calculation**

a) A named calculation can be used to **add a new column** to a table/view in DSV. It is determined by the expression specified.

b) The expression should conform the data source type. For example, if the data source is SQL server, then it should be t-sql statement; if it is from oracle, then pl-sql.

c) There are two common usage of named calculation. The first one is to compute a new column based on the current column, for example, combining the first name and last name into a full name column. Another purpose is to reference another table. Suppose you have a DSV that has product and product category table, and there is a primary key and foreign key relationship, then you can reference column in product category into the product table b using named calculation.

**--named query**

a) A named query creates a new logical table in DSV, or replace an existing table. It can be used to combine columns of multiple table from the data source. A named query is created by using a full select statement. The select statement can contain any SQL code that is compatible to the underlying database, including a where clause, join clauses, group by and so on.

b) It is usage in two common scenarios. The first is to join multiple table from the underlying data source. The second is to filer columns and rows to select only the require data.

Dimensions

--definition:

An SSAS dimension get its data from one or more dimension table in the data source. For example, if the dimension is based on a star schema, then its source is a single dimension table. If it is a based on a snowflake design, then the dimension will typically span multiple dimension tables. In SSAS, we can slice or aggregate a measure according a dimension attributes. For example, we have a production dimension table, in which we have product name, product category and so on. Then we can find the sales amount for a certain product, we can aggregate the sales for product category, as long as there is a relationship between the sales table and product table.

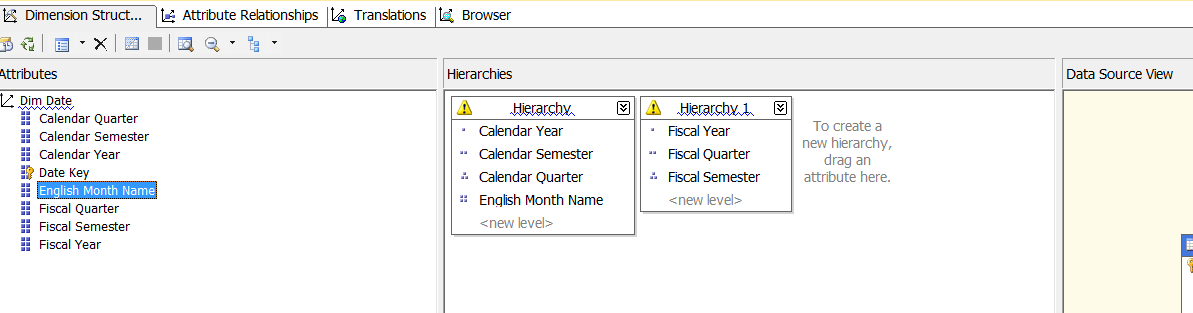
--Database Dimension:

All the dimensions that are created using the Dimension Wizard of the Solution Explorer are treated as database dimensions. In other words, the dimensions which are at Database level are called Database Dimensions. Database dimensions are independent to the cubes; single Database dimension can be used in multiple cubes.

--Cube Dimension:

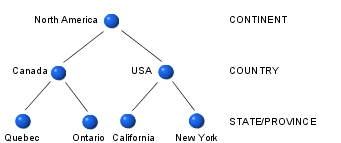
A cube dimension is an instance of a database dimension within a cube. Cube dimension are accessible inside that particular cube, we can’t access cube dimension in another cube.

Component of dimension in SSAS

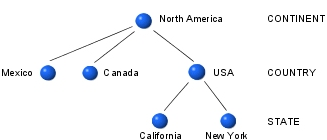


There are three types of hierarchies, including the balanced, unbalance as well the ragged.

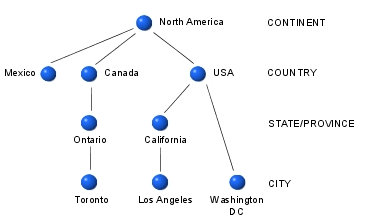
--Balance Hierarchies: In a balanced hierarchy, **each branch of the tree contains the same number of levels**, and **the parent of every member comes from the level immediately above**.



--Unbalanced hierarchies: In an unbalanced hierarchy, **each branch of the tree can descend to a different level,** so it has leaf nodes at more than one level. **The parent of every member comes from the level immediately above.** In the following example, state information is available only for the USA. Mexico and Canada are leaf nodes.



--Ragged hierarchies: In a ragged hierarchy, **the logical parent of at least one member does not come from the level immediately above, but a level higher up.** In the following example, city information for Washington DC relates to the country level, not the state level.



Why hierarchy is good for performance?

Attribute hierarchy is very useful because they let end users to analyze data aggregation by the members of the hierarchy.

Dimension Attribute Relationships

--flexible: this relation is set when you expect the relationship can change over time. For example, a product can change a subcategory over time.

--rigid: used when the relationship doesn’t change over time. For example, a relationship between the quarter and year attribute in a time dimension.

An [Attribute Relationship](https://www.mssqltips.com/sqlservertutorial/2007/creating-a-hierarchy/) is a relationship between various attributes within a Dimension. By default, every Attribute in a Dimension is related to the Key Attribute. Quite often these default Attribute Relationships need to be modified to suit the User Defined Hierarchies and other end user requirements.

There are basically two types of Attribute Relationships:

* ***Rigid***: Attribute Relationship should be set to Rigid when the relationship between those attributes is not going to change over time. For example, relationship between a Month and a Date is Rigid since a particular Date always belongs to a particular Month like 1st Feb 2012 always belongs to Feb Month of 2012. Try to set the relationship to Rigid wherever possible.
* ***Flexible***: Attribute Relationship should be set to Flexible when the relationship between those attributes is going to change over time. For example, relationship between an Employee and a Manager is Flexible since a particular Employee might work under one manager during this year (time period) and under a different manager during next year (another time period).

Why attribute relation is important for performance?

Dimension attribute properties

--KeyColumns

A property of an SSAS Dimension Attribute and it forms the Key (Unique) for the attribute. It can be bound to one or more columns in the underlying database table. When User Defined Hierarchies are created in the dimension (Attribute Relationships defined), setting this property becomes very critical and often requires setting this to a combination of more than one column from the Data Source View. For Example, say you have a Date Dimension and a hierarchy called Calendar Hierarchy (Year -> Quarter -> Month). Now what happens is that, Month gets repeated across different quarters and quarters get repeated across different years making the attribute as non-unique (like January can belong to Q1 of any year and similar Q1 can belong to any year). So to make the attribute unique, KeyColumns for Month should be set to something like Year and Month and similarly for Quarter should be set to Year and Quarter.

NameColumn

A property of an SSAS Dimension Attribute and it is used to identify the column from the underlying Data Source View which provides the name of the attribute which is displayed to the end user by making it more user friendly instead of displaying the Key Column value. For Example, you might have ProductCategoryKey as 1, 2, 3, & 4, and ProductCategoryName as Bikes, Components, Clothing, & Accessories respectively. Now, NameColumn will be set to ProductCategoryName so that user sees them as Bikes, Components etc. even though the data in the background is processed/retrieved using the Key Column values as 1, 2 etc.

Translation

* In SSAS a dimension translation is a language-specific representation of the name of a dimension, the name of an Analysis Services object or one of its members, such as a caption, member, or hierarchy level.
* Translations provide server support for client applications that can support multiple languages.
* Frequently, users from different countries view a cube and its dimensions. It is useful to be able to translate various elements of a cube and its dimensions into a different language so that these users can view and understand the cubes

Cube objects

Cube Structure; Dimension Usage; Calculations; KPI’s (Key Performance Indicators); Actions; Partitions; Aggregations; Perspectives; Translations; Browser

Measure and Measure group

A measure represents a column that contains quantifiable data, usually numeric, that can be aggregated.

Group of Measures having same granularity forms a Measure Group

Dimension Usage

--definition: describes how a dimension is related to a measure group.

--No Relation:

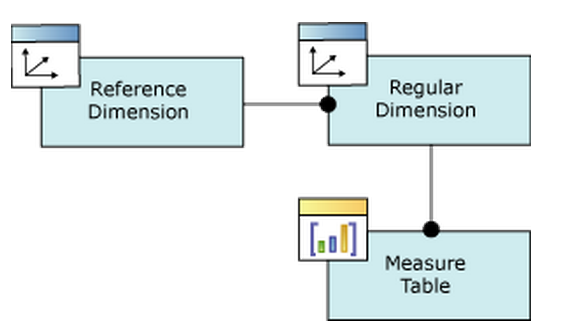
Simply no connection, if there is no relation between a Dimension and Measure Group, there cannot be any aggregations created across that Measure Group

--Regular:

Primary key–foreign key relationship

--Referenced:

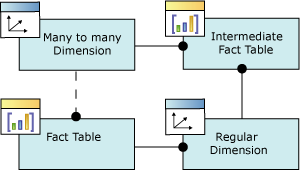
Indirect primary key–foreign key relationship, common in snowflake schema



--Many-to-Many:

Typical many to many relationship

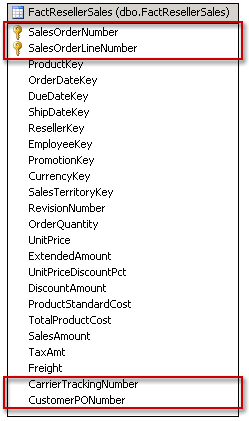
In most dimensions, each fact joins to one and only one dimension member, and a single dimension member can be associated with multiple facts. In relational database terminology, this is referred to as a one-to-many relationship. However, it is frequently useful to join a single fact to multiple dimension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



--Fact:

Frequently referred to as degenerate dimensions

Fact dimensions, frequently referred to as degenerate dimensions, are standard dimensions that are constructed from attribute columns in fact tables instead of from attribute columns in dimension tables. Useful dimensional data is sometimes stored in a fact table to reduce duplication. For example, the following diagram displays the **FactResellerSales** fact table, from the Adventure Works DW Multidimensional 2012 sample database.



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

In theory, you could create a dimension table that uses the same key information as the FactResellerSales table and move the other two attribute columns, CarrierTrackingNumber and CustomerPONumber, to that dimension table. However, you would be duplicating a significant portion of data and adding unnecessary complexity to the data warehouse to represent just two attributes as a separate

**Calculation:**

A calculation is a Multidimensional Expressions (MDX) expression or script that is used to define a calculated member, a named set in a cube in SSAS.

**--calculated member**

A calculated member is a member whose value is calculated at run time using a Multidimensional Expressions (MDX) expression that you specify when you define the calculated member. For example, we can write MDX to calculate the total sales as a sum of reseller sales and internet sales. This calculated member can further be seen as a measure that can be aggregated across dimensions

**--named set**

A named set is a CREATE SET MDX statement expression that returns a set. For example, you can define a named set as the top 50 profitable customers. And used it as a filter to get the 50 most profitable customers.

KPI

In business terminology, a Key Performance Indicator (KPI) is a quantifiable measurement for gauging business success. A KPI is frequently evaluated over time. KPI components are:

* **Value:** An MDX numeric expression that returns the actual value of the KPI.
* **Goal:** An MDX numeric expression or a calculation that returns the target 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.
* **Trend:** An MDX expression that evaluates the value of the KPI over time. The trend MDX expression enables a business user to determine whether the KPI is improving over time or degrading over time.

Value: The KPI value property represents the current value of KPI. This property is typically map to a calculated measure or regular measure.

Goal: This defines the target of KPI

Status: indicates **how KPI compares to the goal**. Its expression should run an integer value of -1 for underperformance, 0 for acceptable performance, or 1 for good performance

Trend: indicates how the KPI value is going over time. Trend should return a value between -1 and 1. It can be the growth rate of underlying KPI over a period of time

Partitions

Partitions are used by SSAS to manage and store **data and aggregations** for a **measure group** in a cube.

Every measure group has at least one partition. Similar to a partition for a table in database, partition a measure group is also related to performance issues. For example, if the measure group has only one partition, the server must to scan the entire partition to find the data to satisfy a query. Say we want to find the total sales in 2014 and if you have 2014 data as a partition, then the server can go to that specific partition to aggregate the data

Partition Storage Mode

A measure group partition can have three storage mode: MOLAP, ROLAP and HOLAP. This three mode various by the layer of data they stored in SSAS server.

A cube can store three type of data, **Meta data, real data and aggregations**. Among which Meta data is always stored on a SSAS server.

For MOLAP, both the cube data and aggregation are stored on the server. This enables fastest query performance since data and aggregation are ready to use in the cube. However, the latency is high new data is only available when the cube is processed. The processing time is fast in this cast because the data is stored efficiently in a compressed format on the server rather than relational database.

For ROLAP, the cube data as well as the aggregation remains in the database. Queries are slow because they must be retrieve from the relational database. The latency is low since you are interact with the database directly.

For HOLAP, as its name suggest, HOLAP is hybrid between MOLAP and ROLAP, it has aggregations on the server but data remains in relational database. It takes less storage and at the same time maintains a reasonable query performance.

Proactive Caching

Proactive Caching is used migrate latency issue for MOLAP as well as HOLAP partition storage mode. It use **silence interval** and **silence override interval** to control caching.

Silence Interval usually set to be 10 seconds. It means SSAS will wait for 10 seconds after receiving a change notification. If no other change comes in this 10 seconds, then the server will build a new version of the cache. If another change happens, then the server will continuous waiting for another 10 seconds

Silence override interval, by default 10 minutes. This is maximum amount of time for waiting. The accumulation waiting amount will no longer than 10 minutes since the first change notification. Even the data is still changing, the server will go head reproduce the cache. In other words, silence override interval is the maximum latency.

Aggregation

Aggregations are **pre-calculated summaries** of data. It improves query response time by preparing the answers before the questions are asked.

--Attribute aggregation options

* Default : SSAS engine will examine the attribute and associated relationships and try to determine whether or not to include that attribute in aggregations using a set of internal rules
* Full : Dimension attribute must be included in every aggregation
* None: Dimension attribute will not be in any aggregation
* Unrestricted: Dimension attribute will be considered without regard to the internal set of rules.

Perspective

It is subset of cube that provide an application focus view of the cube. It controls the visibility of objects in a cute. This includes Dimensions

Attributes

Hierarchies

Measure groups

Measures

Key Performance Indicators (KPIs)

Calculations (calculated members, named sets, and script commands)

Actions

**Cube processing options**

**Process Full:** Processes an Analysis Services object and all the objects that it contains.

**Process Default:** Detects the process state of database objects, and performs processing necessary to deliver unprocessed or partially processed objects to a fully processed state

**Process Data:** Processes data only without building aggregations or indexes. If there is data is in the partitions, it will be dropped before re-populating the partition with source data.

**Process Structure:** Process any structural changes in Cube

**Process Clear:** Drops the data in the object specified and any lower-level constituent objects. After the data is dropped, it is not reloaded.

**Process Index:** Creates or rebuilds indexes and aggregations for all processed partitions. For unprocessed objects, this option generates an error.

**Process Add:** For dimensions, adds new members and updates dimension attribute captions and descriptions. For measure groups and partitions, adds newly available fact data and process only to the relevant partitions.

Dimension Process Options (change add to update)

**Process Full:** Processes an Analysis Services object and all the objects that it contains.

**Process Default:** Detects the process state of database objects, and performs processing necessary to deliver unprocessed or partially processed objects to a fully processed state

**Process Data:** Processes data only without building aggregations or indexes. If there is data is in the partitions, it will be dropped before re-populating the partition with source data.

**Process Clear:** Drops the data in the object specified and any lower-level constituent objects. After the data is dropped, it is not reloaded.

**Process Index:** Creates or rebuilds indexes and aggregations for all processed partitions. For unprocessed objects, this option generates an error.

**Process Update:** Forces a re-read of data and an update of dimension attributes. Flexible aggregations and indexes on related partitions will be dropped.

Roles

Roles are used in Microsoft SQL Server Analysis Services to manage security for Analysis Services objects and data.

* **None:** Members cannot make any modifications to the model schema and cannot query data.
* **Read:** Members are allowed to query data (based on row filters) but cannot make any changes to the model schema.
* **Read and Process:** Members are allowed to query data (based on row-level filters) and run Process and Process All operations, but cannot make any changes to the model schema.
* **Process:** Members can run Process and Process All operations. Cannot modify the model schema and cannot query data.
* **Administrator:** Members can make modifications to the model schema and can query all data.