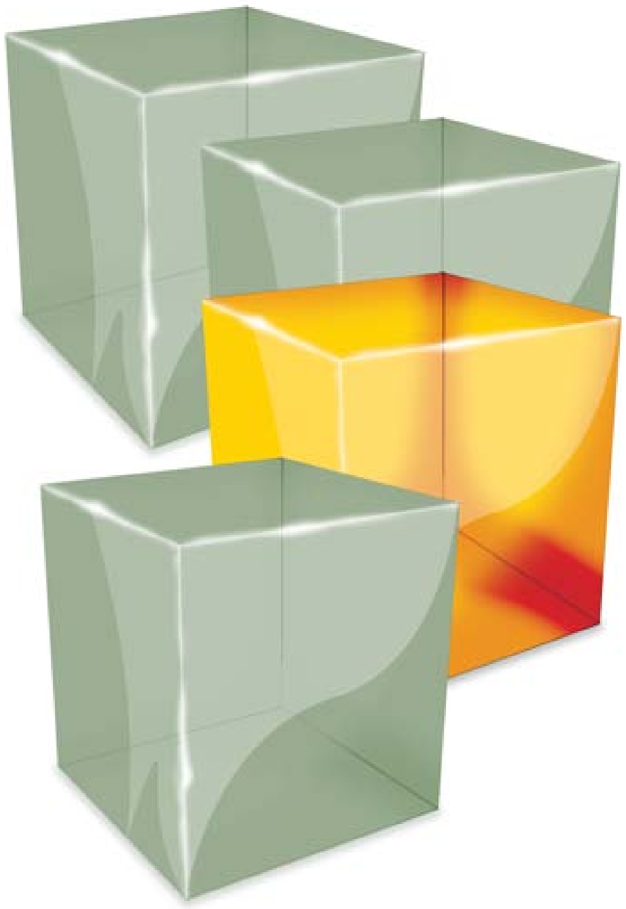
THE OLAP TUTORIAL

- AN INTRODUCTION TO THE MICROSOFT SQL SERVER 2012 ANALYSIS SERVICES



IS5 / IV2014 DATA WAREHOUSING
VERSION A AUTUMN 2012
BY AMIN JALALI

Department of Computer and Systems Sciences (DSV)
Stockholm University (SU)

CONTENTS

INTRODUCTION

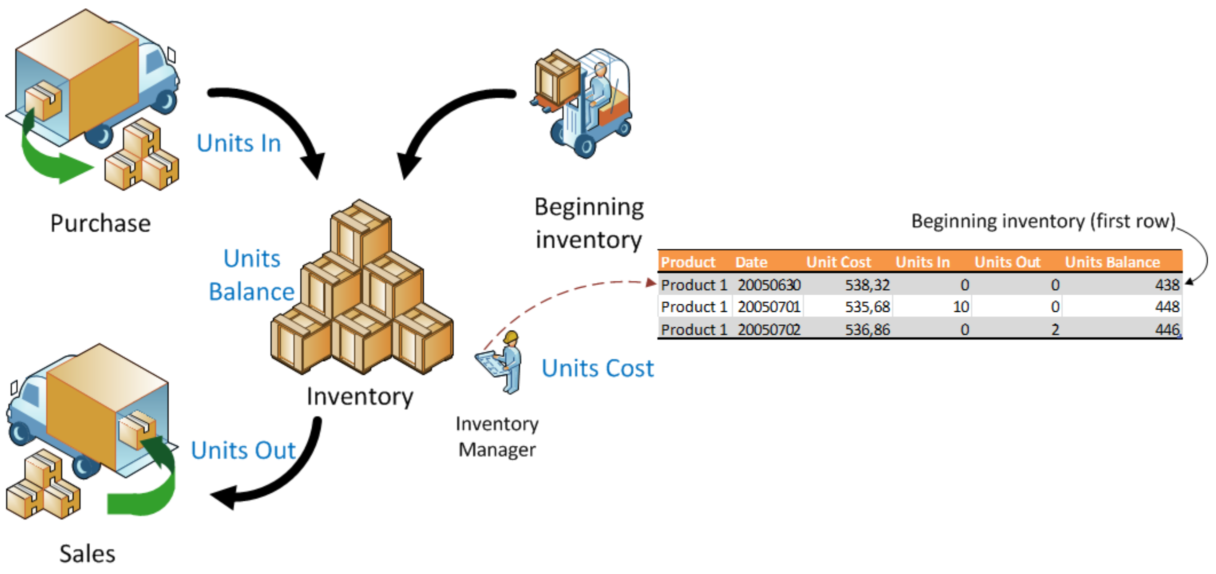
The aim of this document is to make you familiar with Microsoft SQL Server Analysis Services to create an On-Line Analytical Processing (OLAP) project. OLAP technology enables us to store data in multi-dimensional databases, so users can access data quickly. The quick access to data is very critical for data warehouses because of the huge amount of data which are stored in fact tables. Now, it is time to learn how to transform these data into multi-dimensional databases. This is the recommended approach for building data warehouses in practice (1 p. 64).

PURPOSE AND DELIMITATIONS

This document gives an introduction on Microsoft BI/DW tools that are needed to perform the IS5/IV2014 course assignment. Central terminology and concepts of relational databases and data warehouse is a prerequisite and will not be explained in this document. It is also assumed that students are familiar with ETL process.

BUSINESS SCENARIO

This tutorial is designed using a Product Inventory example. It is important to understand the business behind the data warehouse which you intend to design and implement. Therefore, we introduce a product inventory system for which we will design and build the OLAP project briefly.



Product	Date	Unit Cost	Units In	Units Out	Units Balance
Product 1	20050630	538,32	0	0	438
Product 1	20050701	535,68	10	0	448
Product 1	20050702	536,86	0	2	446

Figure 1 Product Inventory System

Inventory is an important part of most product based businesses. Products are traced by measuring their beginning counts, number of purchases and sells. Figure 1 demonstrates an overall picture of an inventory. The 'Units In' and 'Units Out' variables represent the number of products which are purchased and sold. The 'Unit Balance' variable shows the number of products in the inventory. This variable could be calculated as follow:

$$Units\ Balance = Beginning\ inventory + Units\ In - Units\ Out$$

Inventory manager is the person who is responsible of inventory. This person needs to know 'Units In', 'Units Out' and 'Units Balance'. This information will be used to estimate the 'Unit

Cost' which is calculated based on different algorithms. These data are stored in Inventory Systems. For this tutorial, we assume that we received the cleaned and multi-dimensional structured version of data from an ETL team. Thus, we will use these data to design and build an OLAP project.

This data is a part of 'Product Inventory' data mart which is carried out by Microsoft in the 'AdventureWorks DW'[1]. The SSASTutorial database is a subset of data from the 'AdventureWorks DW' database. The SSASTutorial database consists of three tables, i.e. DimDate, DimProduct and FactProductInventory. The first two tables represent data and product dimensions, while the last table represents the product inventory fact table.

[1] The 'AdventureWorks DW' data could be downloaded from http://msftdbprodsamples.codeplex.com.

TOOL EXERCISE

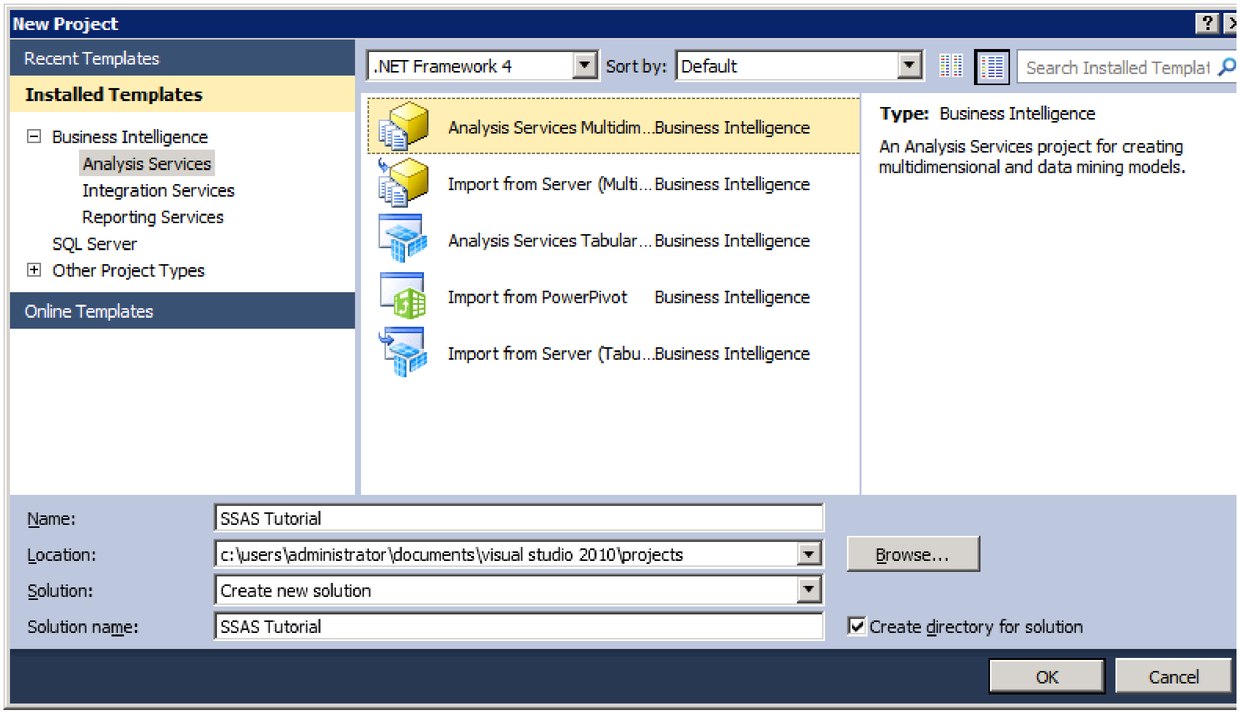
In this section, we make an OLAP project using SQL Server 2012 Analysis Services (SSAS). Then, we learn how to build and deploy the project to the server. Finally, we explore how to utilize Microsoft Excel to analyze OLAP data.

SQL SERVER BUSINESS INTELLIGENCE DEVELOPMENT STUDIO (BIDS)

The SQL Server Business Intelligence Development Studio (BIDS) is Microsoft Visual Studio IDE which enable data warehouse developers to build different data warehouse projects such as Integration Services, Analysis Services and Reporting Services projects. The BIDS environment is intelligently changed based on the project that you choose. In this tutorial we focus on creating an Analysis Services project.

To create an Analysis Services project:

1. Run BIDS from Start → All Programs → Microsoft Visual Studio 2010 → Microsoft Visual Studio 2010.
2. Select File → New → Project to open *'New Project'* window, Figure 2.
3. Select 'Analysis Services' template which is under 'Business Intelligence' in 'Installed template' section at the left side.
4. Select 'Analysis Services Multidimensional and Data Mining Project'.
5. Write 'SSAS Tutorial' in the *Name* field and press *OK*.


New Project
Recent Templates
Installed Templates
Business Intelligence
Analysis Services
Integration Services
Reporting Services
SQL Server
Other Project Types
Online Templates
.NET Framework 4
Sort by: Default
Search Installed Templat
Analysis Services Multidim... Business Intelligence
Import from Server (Multi... Business Intelligence
Analysis Services Tabular... Business Intelligence
Import from PowerPivot Business Intelligence
Import from Server (Tabu... Business Intelligence
Browse...
OK
Cancel

Type: Business Intelligence

An Analysis Services project for creating multidimensional and data mining models.

Name: SSAS Tutorial
Location: c:\users\administrator\documents\visual studio 2010\projects
Solution: Create new solution
Solution name: SSAS Tutorial
[✓] Create directory for solution

Figure 2 New Project Window

As a result, a new SSAS project is created. You can see different parts of a SSAS project in Solution Explorer window at the right section of the BIDS IDE (See Figure 3). It contains the 'SSAS Tutorial' as a project, which consists 'Data Sources', 'Data Source Views', 'Cubes', 'Dimensions' and etc. We describe the first four elements bellow.

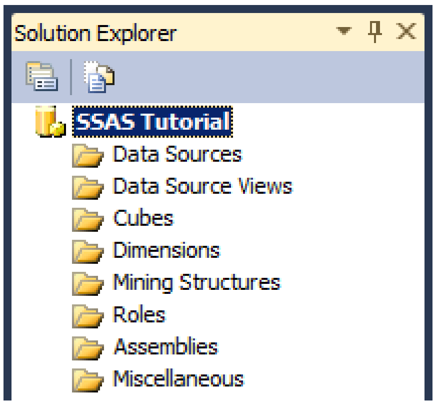


Figure 3 Solution Explorer Window

DATA SOURCES

In this section, we should specify the data source(s) based on which we intend to build the SSAS project. Data Sources are connections to databases from which we load the data into the OLAP storage. In our case, we are going to specify the simplified version of inventory data mart as follow:

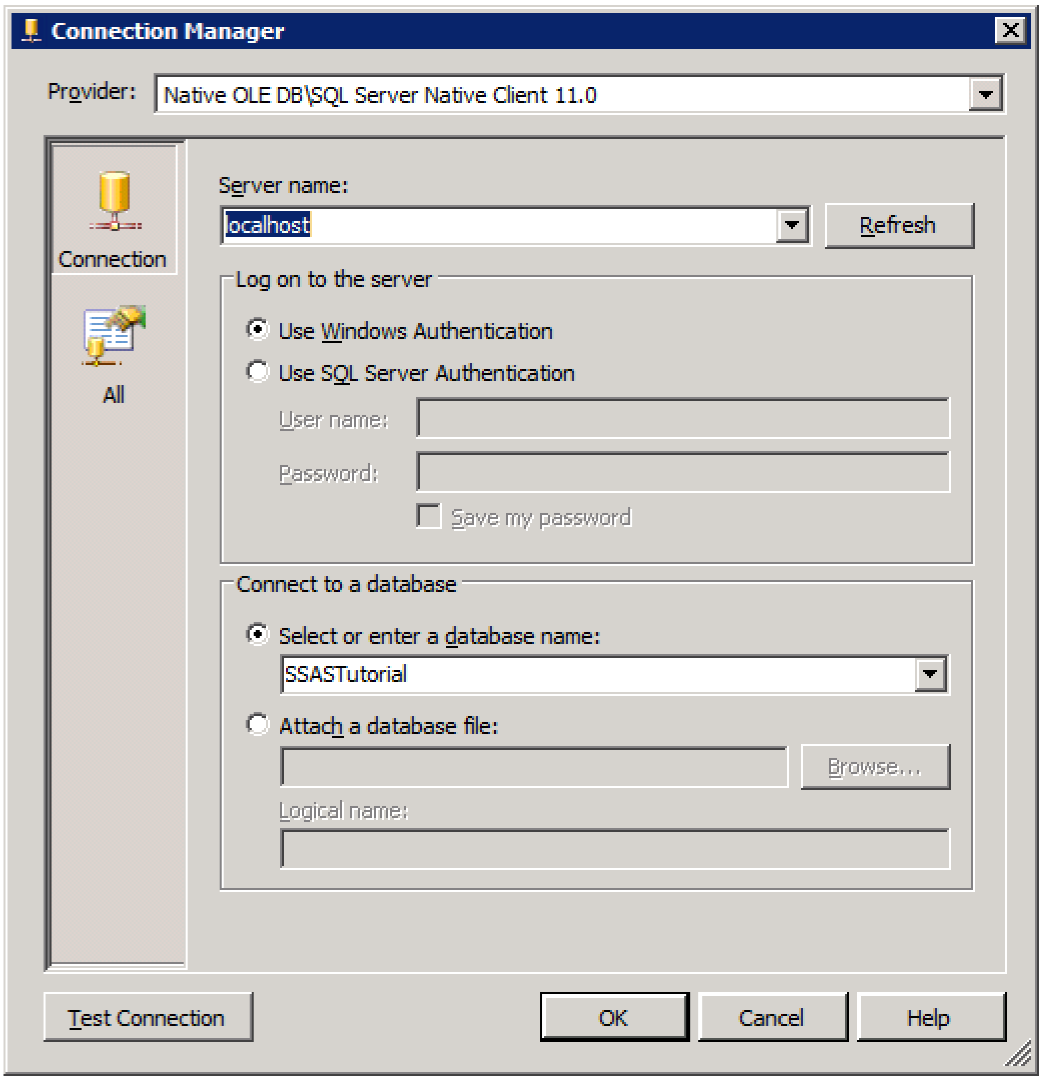
1. Right click on 'Data Sources' folder in Solution Explorer window, and choose 'New Data Source ...' option. The 'Data Source Wizard' window will appear.
2. Click next, and Make sure that 'Create a data source based on an existing or new connection' is selected.
3. Click on New to open the 'Connection Manager' window (see Figure 4).
4. Enter 'localhost' as the server name, and select 'SSASTutorial' from 'select or enter a database name' combobox.
5. Press the 'Test Connection' button to make sure that all things are all right. You should receive a message indicating that 'test connection succeeded'.
6. Press 'OK', and you will come back to the 'Data Source Wizard' window, but you have 'localhost.SSASTutorial' as a selected item in 'Data connections' list.

Now, we have created the data source connection. We should define the security permission that is needed to be employed by SSAS to use this connection. To do that follow the remaining steps:

7. Press next, and select the 'Use the service account' as impersonation information.
8. Click Next. You can see the summary of data source connection in this window. You can also change the name of data source if you want.
9. Click Finish. You will see the data source defined under 'Data Sources' folder in 'Solution Explorer'.

Now, the data source is created and is configured.

X


Connection Manager

Provider: Native OLE DB\SQL Server Native Client 11.0

Server name: localhost

Log on to the server

(●) Use Windows Authentication

( ) Use SQL Server Authentication

User name:

Password:

☐ Save my password

Connect to a database

(●) Select or enter a database name: SSASTutorial

( ) Attach a database file:

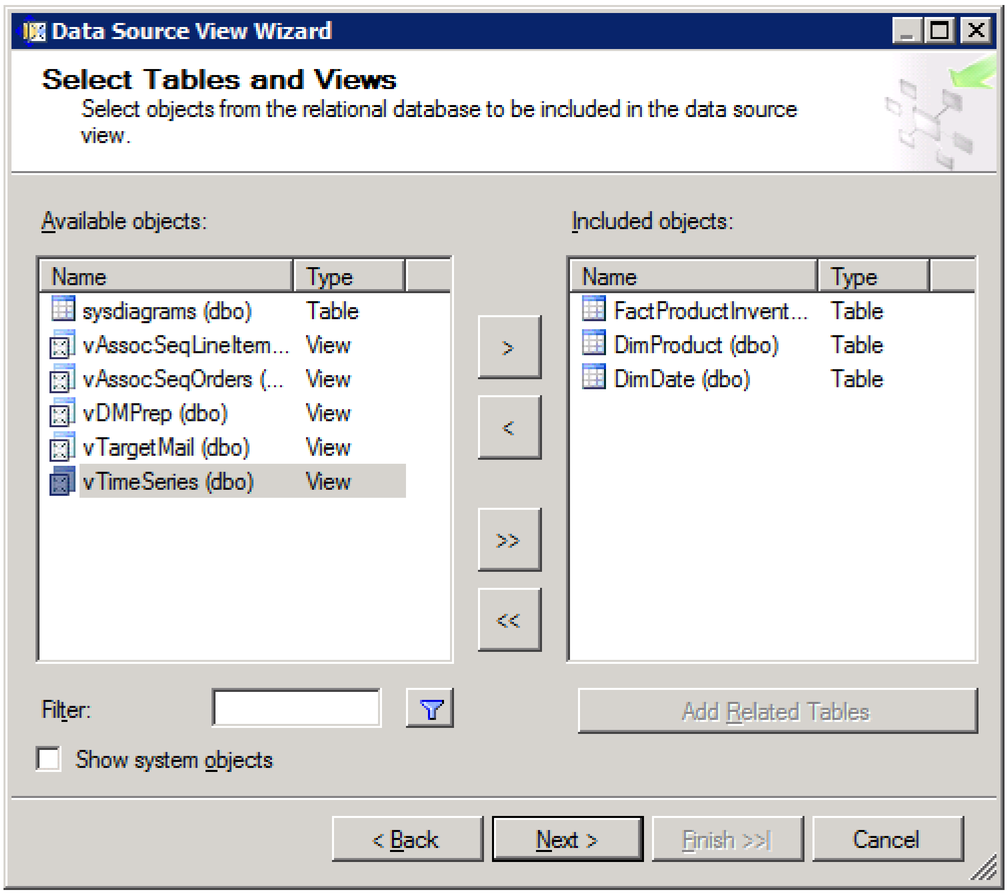
Logical name:

Figure 4 Connection Manager Window

DATA SOURCE VIEWS

Consider the following rewording: “After a data source connection has been specified, it is time to select the tables that will be used in the project. This is done by creating a data source view. In the current project we will define one data source view which contains the tablesDimProduct, DimDate and FactProductInventory. Follow the steps below:

1. Right click on ‘Data Source Views’ folder in Solution Explorer window, and choose ‘New Data Source View …’ Option (The ‘Data Source View Wizard’ window will appear).
2. Click next, and choose the data source that you built in previous section. Then, click next (Here, You can see lists of available objects from the database).
3. Hold Ctrl key and select DimProduct, DimDate and FactProductInventory from the available objects list and add them to the Included objects list by pressing ‘>’ button (See Figure 5). Then Click Next.
4. Click Finish.


Data Source View Wizard

Select Tables and Views

Select objects from the relational database to be included in the data source view.

Available objects:

Name	Type
sysdiagrams (dbo)	Table
vAssocSeqLineItem...	View
vAssocSeqOrders (...	View
vDMPrep (dbo)	View
vTargetMail (dbo)	View
vTimeSeries (dbo)	View

Included objects:

Name	Type
FactProductInvent...	Table
DimProduct (dbo)	Table
DimDate (dbo)	Table

Filter:

☐ Show system objects

Figure 5 Data Source View Wizard Window

When you clicked finish, the BIDS opens the created data source view automatically. You can also open this view by double clicking on the data source object that you created. The data source view editor consists of three parts, i.e. Diagram Organizer, Tables and working space (See Figure 6).

The Diagram organizer in the left pane, is used to categorize related tables. This feature is useful in real projects because you deal with a lot of tables, and you usually want to categorize them in different groups. The 'table section' shows all the tables which are selected in the wizard. The 'working space' is the place that we could add new tables to the 'data source view' or delete some tables from the view. We could also do other things in this working area such as creating 'named calculation'. 'Named calculation' is a column in the table which is calculated based on other columns. We intend to build two named calculations for DimDate table, namely CalendarYear-Semester and CalendarYear-Semester-Quarter. Follow the steps below:

1. Select the DimDate table in working space, and right click on the selected border (See Figure 6). Make sure that the border is selected when you right click on it.
2. Choose 'New Named Calculation …' from the pop-up menu.
3. Enter `CalendarYear-Semester` as the `'Column Name'` and the following SQL as the expression:

```
RIGHT('0000' + CAST(CalendarYear AS NCHAR(4)), 4) + '-' +
CAST(CalendarSemester AS char(1))
```

This expression generates the full semester data through combining the year and its semester. The year and the month are formatted in way to have four and one digits correspondingly.

4. Add another named calculate column called *CalendarYear-Semester-Quarter* and write the following expression:

```
RIGHT('0000' + CAST(CalendarYear AS NCHAR(4)), 4) + '-' +
CAST(CalendarSemester AS char(1)) + '-' +
CAST(CalendarQuarter AS char(1))
```

This expression generates the full quarter data through combining the year, semester and quarter. The year, semester and quarter are formatted in way to have four, one and one digits correspondingly.

5. Save the project.

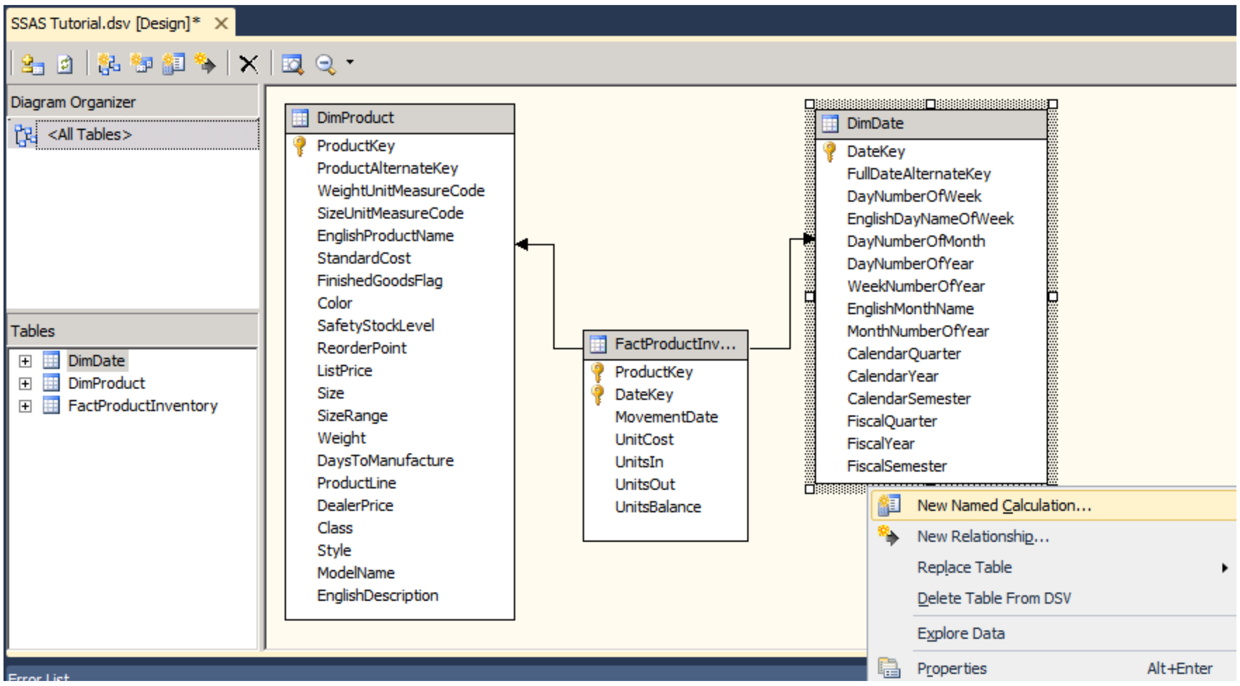


Figure 6 Data Source View IDE

As you might notice, this view is the multi-dimensional star schema of data warehouse. In the next section, we start to design and implement them according to Microsoft OLAP Server (i.e. SSAS) constructions.

DIMENSIONS

In this section, we are going to create two dimensions, i.e. DimDate and DimProduct. In order to create the date dimension, follow these steps:

1. Right click on Dimensions folder in Solution explorer window and select 'New Dimensions …'
2. Click on Next in the welcome screen, and choose 'Use an existing table' option from 'Select Creation Method' screen (This option enables you to design a dimension based on a table. Click Next to continue).
3. In this window, you can choose the data source view and the main table for which you want to create the dimension. Select DimDate as the main table. Make sure that the DateKey column is selected as the Key column. Select FullDateAlternateKey as the 'Name Column' and Click Next.

4. You can select the attributes that you want to use in your dimension here, but it is also possible to add them later. Click Next, then select Finish from the next window.

   It is because you might want to change the dimension structure later, due to change request from a customer.

The dimension editor will be opened automatically (See Figure 7). This window consists of four parts, i.e. *Dimension Structure*, *Attribute Relationships*, *Translations* and *Browser*. *Dimension Structure* enables you to define members of a dimension and existing hierarchies. *Attribute Relationships* enables you to edfine the relations that exists between the dimension's members. *Translations* enable you to add languages which might be used for browsing the dimensions. In this way, you can have multi-language dimension support. *Browsing* enables you to brows the dimension and see how does it look like. We explain how to utilize different parts of this editor in the following sub-sections.

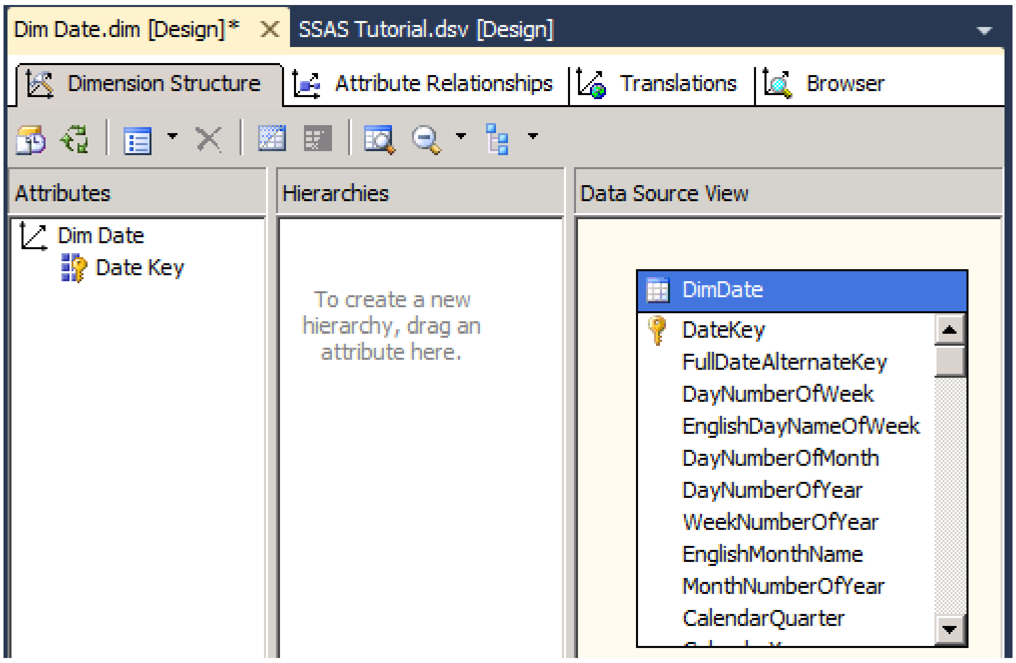


Figure 7 Dimension Editor

ATTRIBUTES

Attributes carry information relevant for a dimension. There are a lot of columns in a data source view that you can add as attributes; however, due to performance considerations, you should only add the attributes which you need for a certain analysis. As you add attributes to the dimension you can also rename them. Follow the steps below:

1. Right click on 'Date Key' attribute (i.e. in the attribute pane on the left) and choose rename option.
2. Rename the 'Date Key' attribute to 'Dim Date'.

Moreover, you can easily drag columns from data source view and drop them in attributes pane of the dimension editor to add attributes to a dimension.

3. Drag and drop the following columns from 'Data Source View' to the Attribute pane: Calendar Year, Calendar Year- Semester, Calendar Year- Semester- Quarter, Fiscal Year,

Fiscal Semester, Fiscal Quarter, Month Number Of Year, Week Number Of Year, English Month Name, English Day Name Of Week.

SSAS provides some features to interact with special kind of dimensions like date dimension. In order to utilize these features, you should introduce your dimension as a special dimension in design time. This setting tells the server that some calculations are valid for this dimension, so we could define special features that could be used for the dimension. For example, we are going to specify year, semester and quarter properties of our dimension to the SSAS. To do that, follow the steps below:

4. Right click on 'Dim Date' dimension in attribute pane and choose properties. Make sure that you selected the dimension, not the Key attributes (because both have the same name now).
5. Change the Type to Time (In this way, we introduce this dimension as a Time dimension to the OLAP Server).
6. Change the type of the following attributes to an appropriate type:
    - Calendar Year → Date.Calendars.Years,
    - Calendar Year- Semester → Date.Calendars.HalfYearOfYear,
    - Calendar Year- Semester- Quarter → Date.Calendars.QuarterOfHalfYear,
    - Dim Date → Date.Calendars.Date,
    - Fiscal Year → Date.Fiscal.FiscalYears,
    - Fiscal Semester → Date.Fiscal.FiscalHalfYears,
    - Fiscal Quarter → Date.Fiscal.FiscalQuarters.

HIERARCHIES

Hierarchies define level of granularity in a dimension. If you want to utilize the power of the OLAP server to store data in multi-dimensional way, you should define hierarchies. We could define two hierarchies for our date dimension, i.e. 'Hierarchy Year Semester Quarter' and 'Hierarchy Fiscal Year Semester Quarter'. The following steps show how to design hierarchies in BIDS.

1. Drag the 'Calendar Year' attribute from the Attributes pane to Hierarchies pan. As you can see, a hierarchy is built which is named Hierarchy.
2. Right click on the Hierarchy title and choose rename option. Change the name to 'Hierarchy Year Semester Quarter'.
3. Drag the 'Calendar Year- Semester' attribute and drop it under the 'Calendar Year' in the hierarchy.
4. Drag the 'Drag Calendar Year- Semester- Quarter ' attribute and drop it under the 'Calendar Year- Semester' in the hierarchy.
5. Drag the 'Dim Date' attribute and drop it under the 'Calendar Year- Semester' in the hierarchy.
6. Your hierarchy should look like Figure 8.

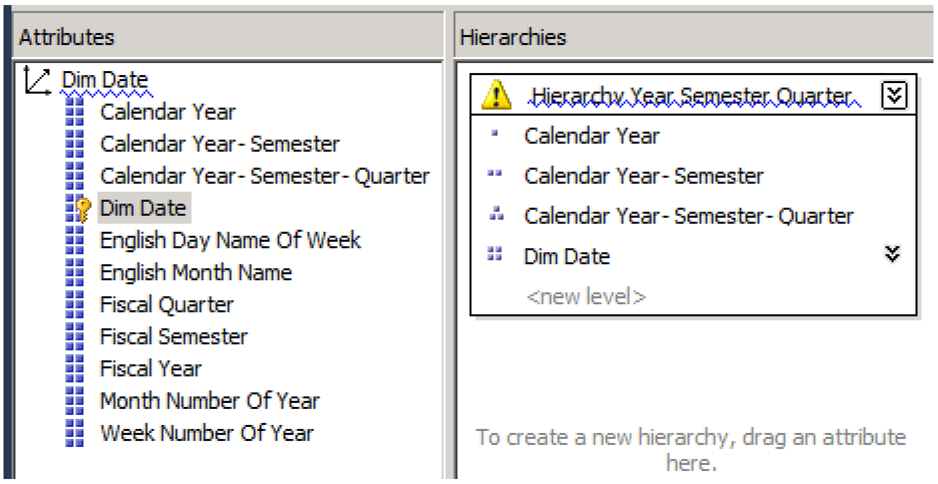


Figure 8 Hierarchy Year Semester Quarter

There are two warnings in the editor, which is shown by blue line under dimension name in attributes pane and hierarchy name in the hierarchy pane. If you bring the mouse pointer over them and stay for a while, the BIDS gives you some hints about them. The first warning indicates that 'Avoid visible attribute hierarchies for attributes used as levels in user-defined hierarchies'. This means that it is better to hide the attributes that are used in a hierarchy, because they could be seen via surfing the hierarchy. To do that, follow these steps:

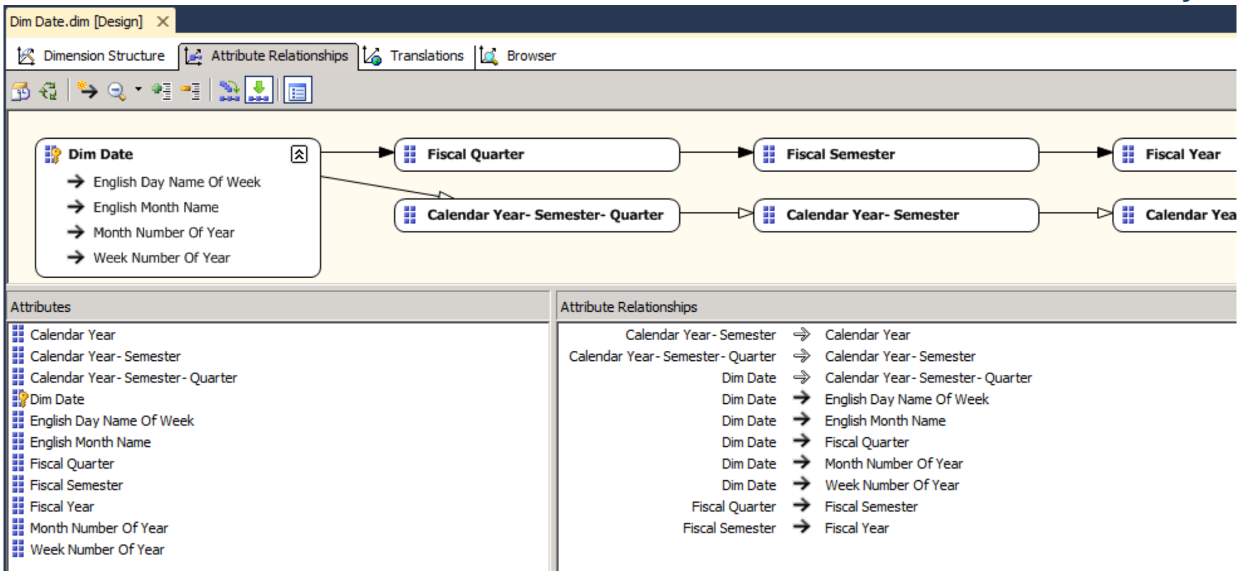
1. Change the AttributeHierarchyVisible properties of 'Calendar Year', 'Calendar Year-Semester', 'Calendar Year- Semester- Quarter' and 'Dim Date' attributes to False. So the first warning should be gone.

The second warning indicates that 'Attribute relationships do not exist between one or more levels of this hierarchy. This may result in decreased query performance'. We will see what this means and how we should solve it in the next section (Attribute Relationships), but first we are going to create our second hierarchy 'Hierarchy Fiscal Year Semester Quarter' .

2. Drag the 'Fiscal Year' attribute to the hierarchy pane, so a new hierarchy is created.
3. Rename the hierarchy to 'Hierarchy Fiscal Year Semester Quarter'.
4. Add the following attributes to the hierarchy: 'Fiscal Semester', 'Fiscal Quarter' and 'Dim Date'.
5. Change the AttributeHierarchyVisible properties of 'Fiscal Year', 'Fiscal Semester' and 'Fiscal Quarter' to false.

ATTRIBUTE RELATIONSHIPS

This section enables us to define the relation between attributes of a dimension. Each attribute in a dimension could be related to other attributes in different ways and this tab enables such definition. This definition will improve the performance of the SSAS. You can define the relationships between attributes of a dimension in the 'Attribute Relationships' tab. This tab contains three parts, i.e. Attributes, Attribute Relationships and Working. Attributes section shows the attribute of the dimension, and the 'Attribute Relationships' section provides you with facilities to define and edit these relationships. You can also see the graphical representation of relationships in the working area. This area also enables you to define and change the relationships. Your screen is different with this figure, but you will end up with the same screen after following the steps below.



Attributes

- Calendar Year
- Calendar Year- Semester
- Calendar Year- Semester- Quarter
- Dim Date
- English Day Name Of Week
- English Month Name
- Fiscal Quarter
- Fiscal Semester
- Fiscal Year
- Month Number Of Year
- Week Number Of Year

Attribute Relationships

Calendar Year- Semester	→	Calendar Year
Calendar Year- Semester- Quarter	→	Calendar Year- Semester
Dim Date	→	Calendar Year- Semester- Quarter
Dim Date	→	English Day Name Of Week
Dim Date	→	English Month Name
Dim Date	→	Fiscal Quarter
Dim Date	→	Month Number Of Year
Dim Date	→	Week Number Of Year
Fiscal Quarter	→	Fiscal Semester
Fiscal Semester	→	Fiscal Year

Figure 9 Attribute Relationships

1. Delete relationships between 'Dim Date' and 'Fiscal Semester', 'Fiscal Year', 'Calendar Year- Semester' and 'Calendar Year' in the 'Attribute Relationships' section.
2. Right click on empty area in working space and select 'New Attribute Relationship …' from the menu.
3. Choose the 'Calendar Year- Semester- Quarter' as the source attribute and 'Calendar Year- Semester' as the related attribute.
4. Change the relationship type to 'Rigid'. This means that the relation between 'Calendar Year- Semester- Quarter' and 'Calendar Year- Semester' will not be changed over time, while Flexible option means that it could be changed overtime. For example, you might define an organizational chart dimension as flexible, because the position of people changes over time.
5. Click Ok.

Do above steps to define the following attribute relationships as well:

- Calendar Year- Semester → Calendar Year (rigid)
- Fiscal Quarter → Fiscal Semester (flexible)
- Fiscal Semester → Fiscal Year (flexible)

If you click on one of these relationships, you can see that you can change the cardinality of the relationship from Many to One. Do not change the property since we do not have such a relationship here.

Edit the following relationships and change their Relationship Type setting to 'Rigid':

- Dim Date → English Day Name Of Week
- Dim Date → English Month Name
- Dim Date → Month Number Of Year
- Dim Date → Week Number Of Year

Now you should have the same screen as Figure 9. If you go back to 'Dimension Structure' tab, you will see that the warnings have disappeared. This is due to the fact that now the relationships are defined.

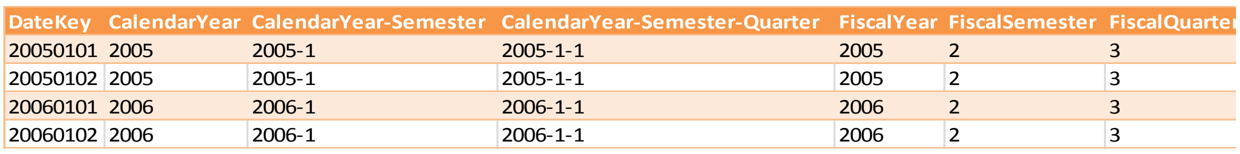
DEPLOYING AND BROWSING

It is time to deploy our project to the OLAP server and see how our dimension looks like. To do so, first you must check the configuration of deployment.

- Right click on the 'SSAS Tutorial' project and select properties.
- Choose Deployment from the left hand side.
- Make sure that the Server is set to localhost, and the database is set to 'SSAS Tutorial'.
- Right click on project name and click Deploy.

You should face some errors. One of these errors indicates that 'An error occurred while the 'Calendar Year- Semester- Quarter' attribute of the 'Dim Date' dimension from the 'SSAS Tutorial' database was being processed'. This is due to structure of the data for which we defined hierarchies, the current structure does not comply to the data. Let's consider what causes this error and how we could solve it.

Right click on the DimDate dimension in data source view part of dimension designer and click on 'Explore Data'. A window will open which shows some part of the relational data of the date dimension. Here you can explore how the data in the relational source table looks like. We show some rows of these data to explain the problem (See Figure 10).



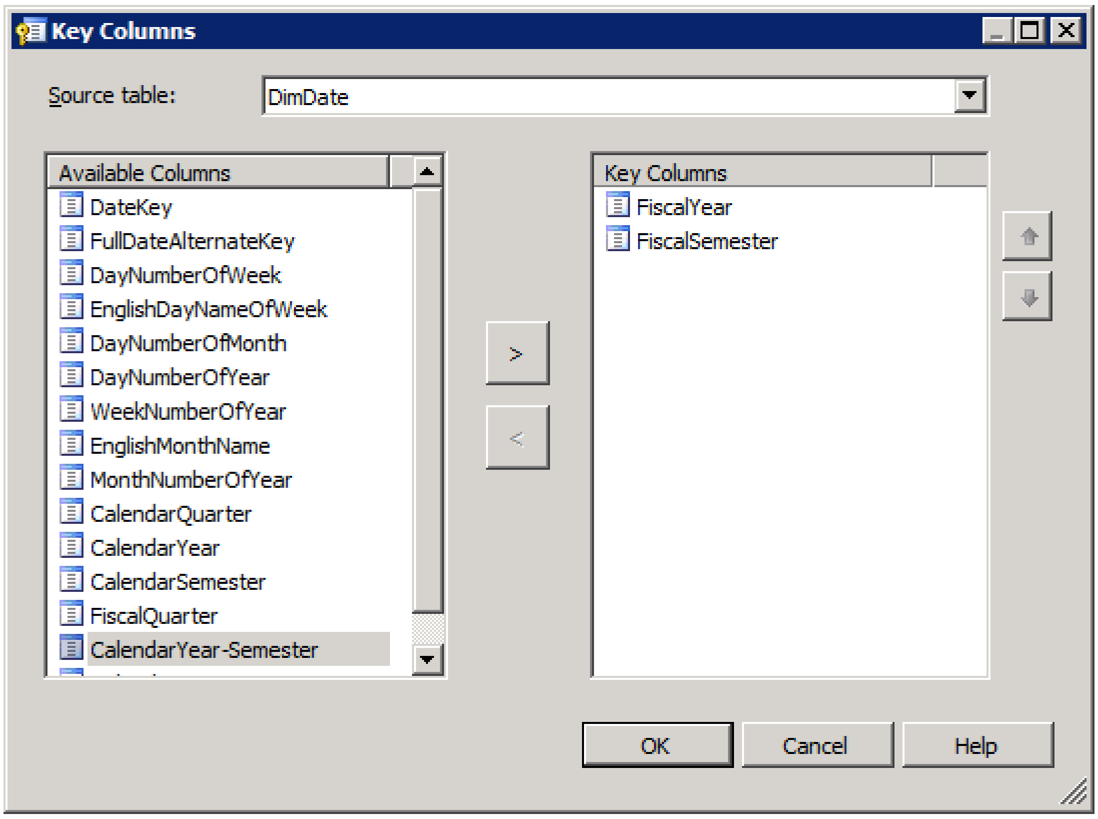
DateKey	CalendarYear	CalendarYear-Semester	CalendarYear-Semester-Quarter	FiscalYear	FiscalSemester	FiscalQuarte
20050101	2005	2005-1	2005-1-1	2005	2	3
20050102	2005	2005-1	2005-1-1	2005	2	3
20060101	2006	2006-1	2006-1-1	2006	2	3
20060102	2006	2006-1	2006-1-1	2006	2	3

Figure 10 Data Example

We defined the *[Hierarchy Year Semester Quarter]* to have a one to many relationships between *Calendar Year, Calendar Year- Semester, Calendar Year- Semester- Quarter and Dim Date* sequentially. As you can see in the first row of Figure 10, we have a one to many relationships between *CalendarYear* and *CalendarYear-Semester* (2005 -> 2005-1). Therefore, the data has consistency with the definition of hierarchy. However, if you consider the other dimension, *[Hierarchy Fiscal Year Semester Quarter]*, the data violates the defined relationship. For example, we could not reach the FiscalSemester from FiscalYear. If you consider the first row of Figure 10, we could not say the FiscalSemester with value 2 is related to which FiscalYear! Thus, there is not a pure one-to-many relationship between 'Fiscal Year' and 'Fiscal Semester' data.

This means that the data relationship is not one-to-many according to what is specified in the design, so how could we solve this problem? We could solve the problem by adding the calculate members in Data Source View, like what we did for 'Hierarchy Year Semester Quarter' hierarchy. However, it is not wise to assign extra memory to solve this problem. If we add the year key to the Fiscal Semester key, the problem will be solved. The first solution added a field to the DSV, and stored it in cube with two other keys. However, the second solution just uses those keys and does not store the extra key. Therefore, it will use less memory space. It means that we should have:

- Fiscal Year:2005 → Fiscal Semester:2005,2
- Fiscal Year:2006 → Fiscal Semester:2006,2



Source table: DimDate

Key Columns: FiscalYear, FiscalSemester

Figure 11 Key Columns Window

In this way, the relationship will not be violated. In other word, you can precisely relate a FiscalYearSemester to a FiscalYear. To do that, follow these steps:

- Right click on 'Fiscal Semester' attribute in attributes pane of date dimension, and select properties.
- Choose ellipsis button from keyColumns section.
- You can see that 'Fiscal Semester' Column is selected as the key column. Add the 'Fiscal Year' column to have both as key columns. Move the 'Fiscal Year' column up (See Figure 11).
- Click Ok.
- Choose FiscalSemester as the NameColumn for Fiscal Semester Attribute as well.

Repeat these steps for Fiscal Quarter attribute. Set the key to the combination of 'Fiscal Year', 'Fiscal Semester' and 'Fiscal Quarter'. Set the NameColumn to 'Fiscal Quarter'. Now you can deploy the project without any errors. When the deployment is finished successfully, you can choose the brows tab to explore the dimension.

PRODUCT DIMENSION

We should also create the product dimension before designing the cube.

- Create this dimension with the following attributes: Dim Product (Key attribute), Class, Color, Days To Manufacture, Dealer Price, English Description, English Product Name, List Price, Model Name, Product Line, Size and Size Range.
- Moreover, create a hierarchy called 'Hierarchy SizeRange Size' with these levels: Size Range → Size → Dim Product. Deal with warnings, and define attributes relationships in way like Figure 12.

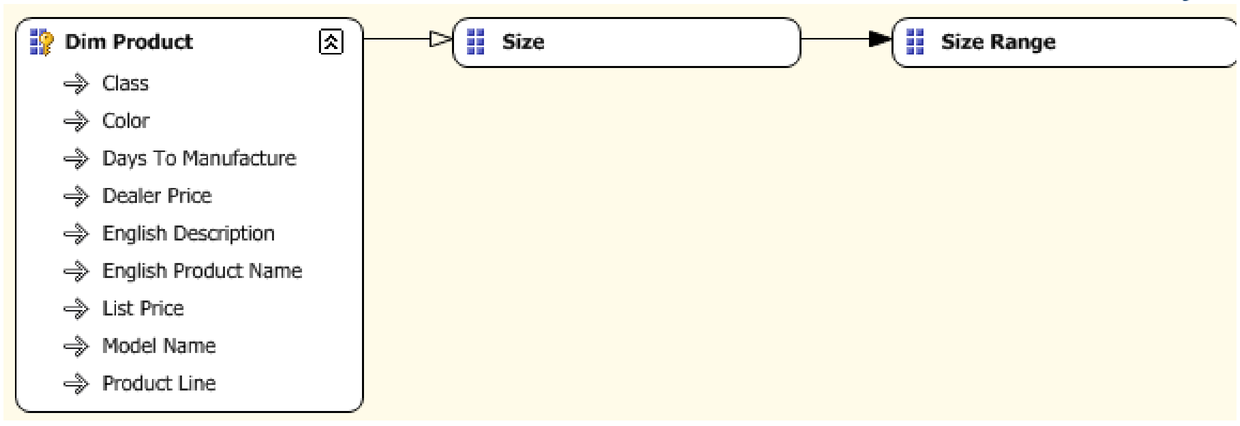


Figure 12 Product Attribute Relationships

Deploy and brows the dimension.

CUBES

Cubes are the most important part of an OLAP project. Cubes are used to organize and physically store data in order to facilitate a quick access and analysis of it. In order to create a cube in SSAS, you can follow the steps below:

- Right click on Cubes folder in Project Explorer window and choose 'New Cube …'
- Press Next button in the welcome screen
- Select 'Use existing tables' to indicate how you would like to create your cube and press Nest button
- Choose 'SSAS Tutorial' as your data source view and select 'FactProductInventory' as the measure group tables and press Nnext
- Select Unit Cost, Units In, Units Out and Units Balance from the measure and press Next
- Choose 'Dim Date' and 'Dim Product' as dimensions which should be included in the cube and press next
- Review the work, and press Finish.

Your cube is built now. The cube designer opens it automatically. This designer consists of several tabs (See Figure 13), from which we will explain Cube Structure, Dimension Usage, Calculations, KPIs, Actions and Browser tabs.

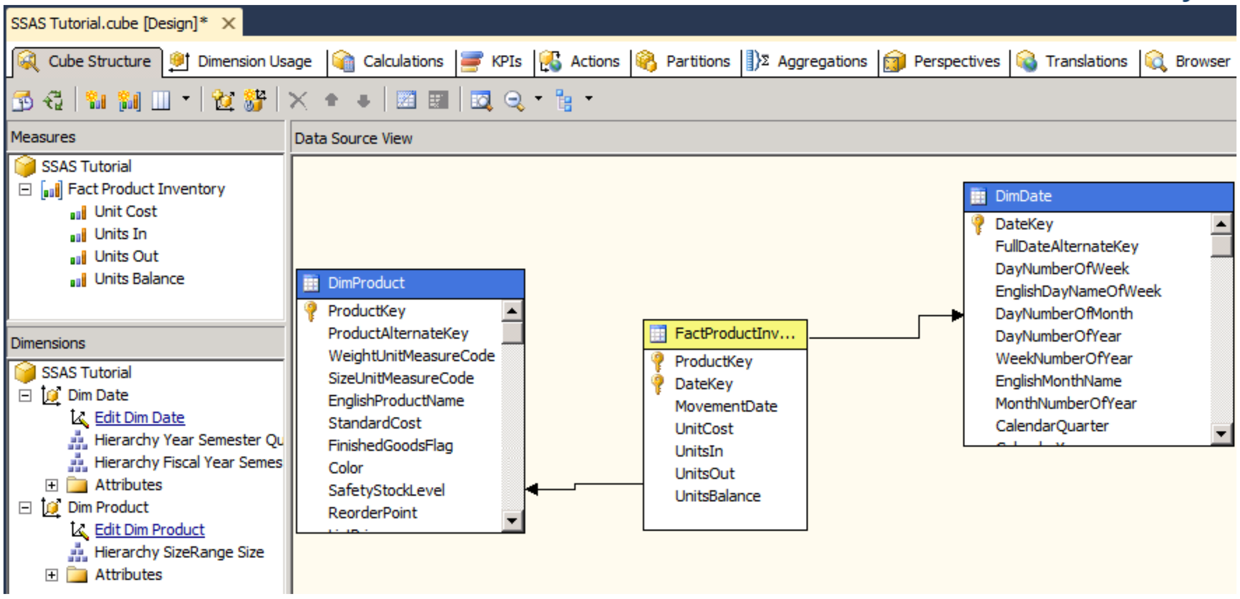


Figure 13 Cube Structure tab of Cube Designer

CUBE STRUCTURE

The cube structure shows the dimensions and fact tables which are utilized in the cube. The facts are called measures, and they are grouped based on their usage in measure groups. SSAS creates a measure group for each fact table by default, but you can change it at any time. You can change the properties of facts to determine how they should be configured in the cube. Each measure has some properties. One of these properties is AggregateFunction which indicates how measures should be aggregated in different level of dimensions. SSAS uses Sum function by default. In our project, the ‘Units In’ and ‘Units Out’ measures are additive facts, so they could be summed up in different levels. However, the ‘Unit Cost’ and ‘Units Balance’ are non-additive facts and could not be summed up. We are interested to reflect the latest value for these measures, so we are going to change these functions as bellow:

- Right Click on Unit Cost measure (in the Measures window) and choose properties
- Change AggregateFunction to Last Nonempty
- Repeat the two steps above for Units Balance

DIMENSION USAGE

This tab is used to configure the relation between dimensions and measure groups which are added in the cube structure tab. As you can see in Figure 14, date dimension is related to a measure group using Dim Date measure. Indeed, the Dim Date acts as a key that joins the dimension to the fact table. This indicates that ‘Dim Date’ measure is the key measure which relates your dimension to the measure group (Fact table). There are different kinds of relationships which could be defined here, but it is out of the scope of this tutorial to explain them all. If you are interested you can read more about in Microsoft msdn[2].

[2] http://msdn.microsoft.com/en-us/library/ms178448.aspx

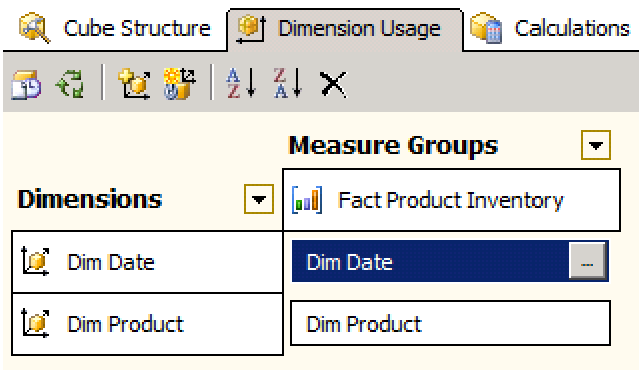


Figure 14 Dimension Usage tab

CALCULATIONS

In this section, you can define some, so called, calculation members which could help users to get more from their OLAP data. This tab consists of three parts including 'Script Organizer', 'Calculation Tools' and 'Script Editor' (See Figure 15). The 'Script Organizer' could be used to organize scripts. You can create a new calculated member, named set or script commands using this window.

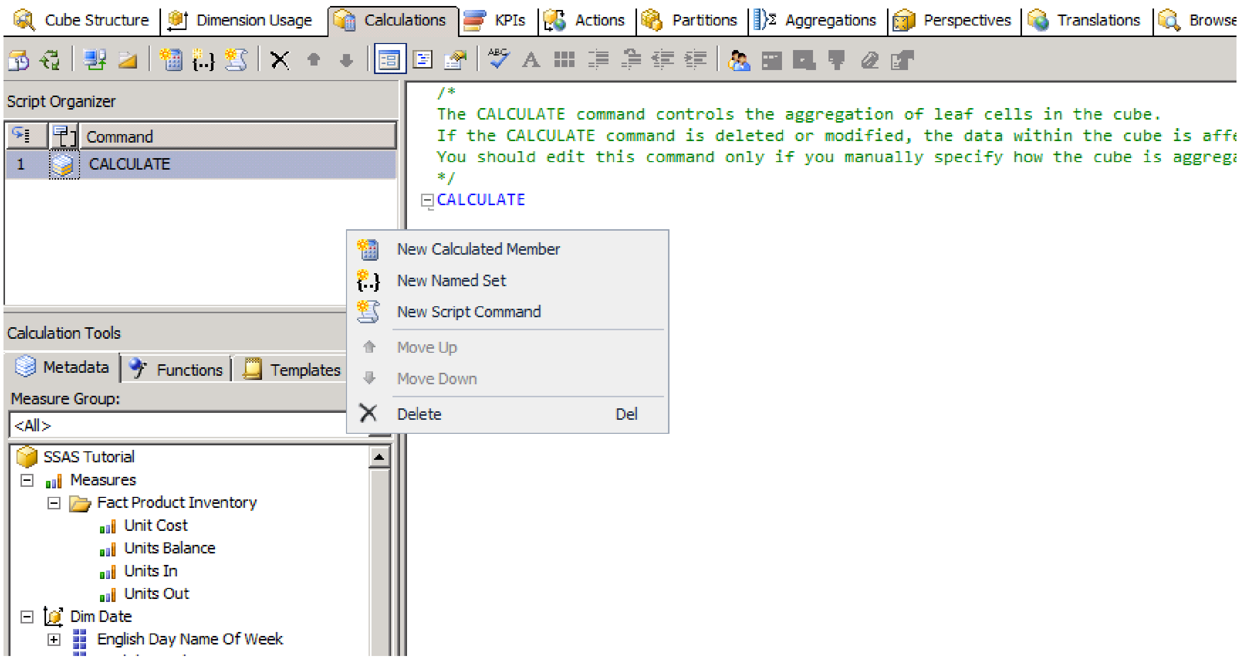
Assume for example, that users would like to know the total value of their products in inventory. This value could be calculated by multiplying unit cost and units balance. Follow the steps below to create a calculated member:

- Right click on 'Script Organizer' window and choose 'New Calculated Member' option.
- Change the name of created member to 'TotalValue'
- Select Measures from the 'Parent hierarchy' combo box.
- Enter the following code as expression: [Unit Cost] * [Units Balance]
- Set the 'Associated measure group' to 'Fact Product Inventory'

This was a simple calculated member that we defined. BIDS helps us to build more sophisticated and useful calculated members by providing templates for functions and calculated members in the 'Calculation Tools' window.

This section could be used for looking at the 'cube structure', using pre-defined functions and customizing existing templates. You might realize that there is an error in Metadata section of this window instead of cube structure that could be seen in Figure 15. This is due to the fact that the metadata section read the metadata of cube from the server, but you have not deployed your cube, so there is no metadata in the OLAP server. To do that you should right click on the project, and press Deploy. After the successful deployment, you can press the 'Reconnect' button in the menu bar to refresh the window. You should be able to see the metadata of dimensions and measures now.

The 'Calculation Tools' contains also two other tabs which could be used when defining new calculated members. We explain how to use the template tab using an example. Imagine that we want to enable users to know about the growth of units balance from period to the next period based on the fiscal year of a company. To define such a calculation, we should write some expression to calculate it.



```
/*
The CALCULATE command controls the aggregation of leaf cells in the cube.
If the CALCULATE command is deleted or modified, the data within the cube is affe
You should edit this command only if you manually specify how the cube is aggrega
*/
CALCULATE
```

Figure 15 Calculations tab

Follow these steps to define such calculation:

- In Templates tab of 'Calculation Tools' window, expand 'Time Series' and double click on 'Period over Period Growth', so a new calculated member is created.
- Change the name to '[Units Balance Period over Period Growth]'.
- In Expression section, apply these changes:
  - Change '[<<Target Dimension>>].[<<Target Hierarchy>>]' to '[Dim Date].[Hierarchy Fiscal Year Semester Quarter]'
  - Change '[<<Target Dimension>>].[<<Target Hierarchy>>].[<<Target Level>>]' to '[Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year]'
  - Change '[Measures].[<<Target Measure>>]' to '[Measures].[Units Balance]'
  - Change '<<Number of Periods>>' to 1
  - Now your script should look like the script in Figure 16.
- Change the format string to "Percent".
- Change the associated measure group to 'Fact Product Inventory'.
- Expand the 'Color Expression' box, and enter this expression as 'Fore Color' value:

```
IIF([Measures].[Units Balance Period over Period Growth] < 0, 255 , 0)
```

This expression check if the value of this measure is less than zero or not. If it is negative, then it shows it with red color.

You can deploy the project to see these calculated members in brows tab. Drag the 'Units Balance Period over Period Growth' calculated member which is located in measures list in brows tab and drag it into the result. As you can see, the 'NA' is returned. If you drag the Fiscal hierarchy and drop it in the result section, you will see the growth. However, values are neither colored nor formatted! This is a bug in browsing section of BIDS. However, in client application like MS Excel it still works fine. We will consider them later when we learn how to explore the OLAP data in MS Excel (See Using Microsoft Excel section).

```
Case
// Test for current coordinate being on (All) member.
When [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember.Level Is
     [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[(All)]

Then "NA"

// Test to avoid division by zero.
When IsEmpty
     (
       (
         ParallelPeriod
         (
            [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year],
            1,
                  [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember
         ),
        [Measures].[Units Balance]
        )
     )
Then Null

Else (
       ( [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember,
            [Measures].[Units Balance] )
          -
       (
         ParallelPeriod
         (
            [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year],
            1,
               [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember
         ),
         [Measures].[Units Balance]
       )
        )
        /
     (
       ParallelPeriod
       (
          [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year],
          1,
             [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember
       ),
       [Measures].[Units Balance]
     )
End// This expression evaluates the difference between the value of the numeric
```

Figure 16 Customized Calculated member template

This expression calculates the difference of Units balance for this quarter and compares it to the same quarter in previous year. As it could be seen, if the user does not expands the quarter level, it returns 'NA' since no quarter is selected. Nor does it return any value if the current Unit Balance is empty, because we could not divide a value by zero (The empty will be considered as zero). In the last part of the MDX, the expression divides differentiations of current member with the same value for previous year. Then, it divide the differentiation by the current member. Therefore, the growth percentage will be calculated.

As you might have noticed, there is a default calculated member which is named CALCULATE in the Script Organizer tab. This is an important calculated member, because it tells SSAS to process other calculated members. If you delete this member, none of your calculated members will be processed.

KEY PERFORMANCE INDICATORS (KPIs)

Key Performance Indicators (KPIs) could be defined in SSAS to represent the organization's status and trends based on pre-defined goals (2). These KPIs can be defined using expressions that will be used by the OLAP server to calculate them based on fact data. Therefore, stakeholders could always warned about the business status, critical performance status of the business and etc. The KPI tab in the Cube Editor is used to define and edit KPIs. This editor is exactly like the Calculations Editor. In this section, we design a simple KPI. Follow the steps below:

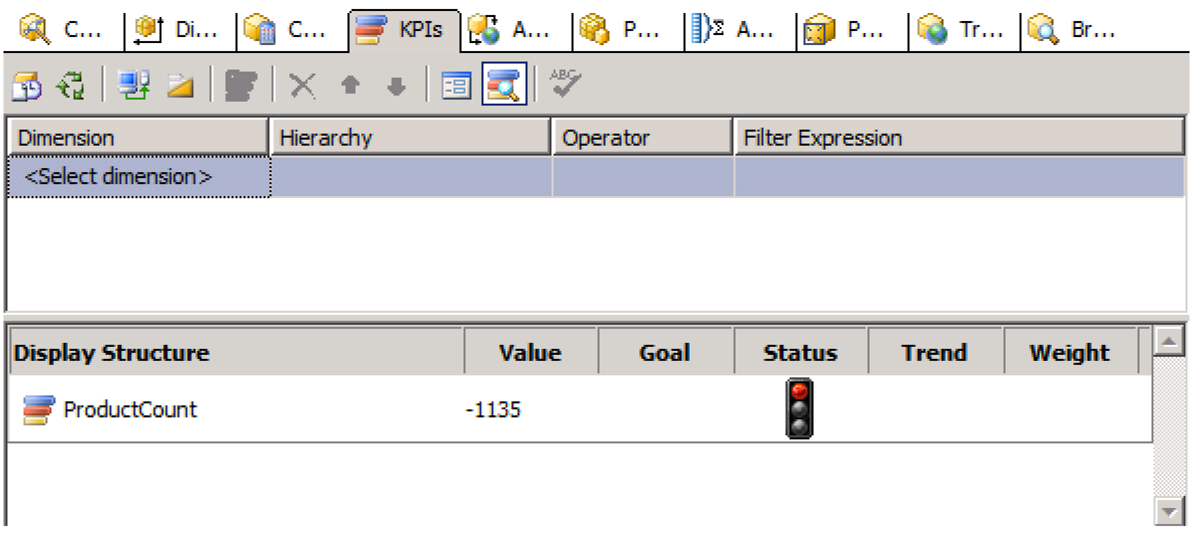
- Right click inside the 'KPI Organizer' window and press 'new KPI'
- Enter 'ProductCount' as the name of the new KPI
- Set the 'Associated measure group' to 'Fact Product Inventory'
- Set the 'Value Expression' to '[Measures].[Units In] - [Measures].[Units Out]'
- Set the 'Status Indicator' to 'Traffic Light'
- Set the 'Status expression' as following:

```
CASE
WHEN [Measures].[Units In] - [Measures].[Units Out] > 10
THEN 1
WHEN [Measures].[Units In] - [Measures].[Units Out] <= 10
AND
[Measures].[Units In] - [Measures].[Units Out] >= 0
THEN 0
ELSE -1
END
```

This expression returns:

- '1' if the difference between 'Units In' and 'Units Out' is greater than 10. This means that we have enough items in the stock.
- '0' if the difference between 'Units In' and 'Units Out' is between 0 and 10. This means that we still have enough items in the stock, but we should make an order for more items.
- '-1' if the difference between 'Units In' and 'Units Out' is less than 0. This means that we sold more than we bought. Therefore, we should buy products.

Now you have successfully created a KPI. Deploy the project, and press the 'Browser View' button from the KPIs menu bar, see Figure 17 (Note that this is a different Browser button than the Browser tab in the Cubes window). You can see the KPI visually. Later, we will see how we could use KPIs in client application like MS Excel.



Dimension	Hierarchy	Operator	Filter Expression
<Select dimension>			

Display Structure	Value	Goal	Status	Trend	Weight
ProductCount	-1135		●		

Figure 17 KPI Browser

ACTIONS

Actions enable users to act and make a decision when they are browsing data. “For example, suppose that a user browses an inventory cube and discovers a low level in the inventory of a certain product. He can use actions to trigger a supply chain system procurement order for the product” (2). The action editor is very similar to KPI and Calculation editor.

We look at how to define a simple action using an example. Imagine that our users want to Google models of products. To define such an action you can follow these steps:

- Right click on ‘Action Organizer’ window and choose ‘New Action’.
- Change the name of action to ‘SearchModelName’
- Change ‘Target type’ to ‘Attribute members’
- Set the ‘Target object’ to ‘Dim Product.Model Name’
- Write the following expression for ‘Action expression’. This expression produces a URL to google the product name.

```
"http://www.google.com/search?q=" + [Dim Product].[Model Name].CurrentMember.
MEMBER_CAPTION
```

It seems that BIDS 2010 does not support actions when browsing cubes in BIDS. However, client applications like MS Excel can utilize actions. We will see how to use this action in MS Excel in the next section.

DEPLOYING AND BROWSING

When you use the deploy option, the OLAP schema is partially processed and the data will be deployed totally. Before we start considering how to use client applications to access OLAP server, we are eager to process the data. To process the OLAP structures follow these steps:

- Right Click on your project and choose ‘Process …’ option. This option processes the OLAP schema and applies the changes to the server.
- If a window appears which asked ‘The server content appears to be out of date. Would you like to build and deploy the project first?’ choose Yes.
- If another window appears which ask about overwriting the database, choose yes.
- In the process dialogue, press ‘Run …’

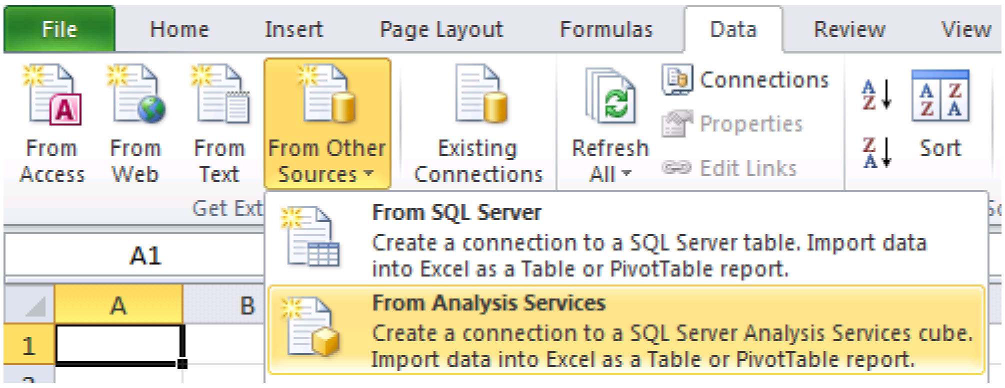
- When finished, close the two open windows.

You have successfully built, deployed and processed the OLAP data.

USING MICROSOFT EXCEL

There are various applications which could connect to SSAS server to retrieve information. One widely used application is MS Excel. In this section, you will learn how to use MS Excel to get information from the SSAS. Follow the steps below to connect MS Excel to SSAS:

- Run MS Excel. Choose 'From Other Sources' option from the Data menu.
- Select 'From Analysis Services' option from the list (as shown in Figure 18).

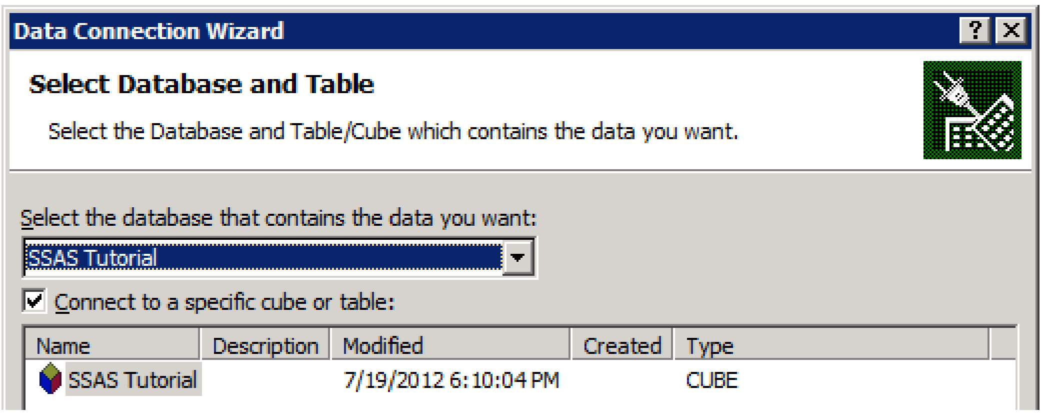


From SQL Server
Create a connection to a SQL Server table. Import data into Excel as a Table or PivotTable report.

From Analysis Services
Create a connection to a SQL Server Analysis Services cube. Import data into Excel as a Table or PivotTable report.

Figure 18 Connecting MS Excel to SSAS

- Enter 'localhost' as server name, and press Next.
- Select 'SSAS Tutorial' as database, and make sure that the 'Connect to a specific cube or table' checkbox is selected. Then select the 'SSAS Tutorial' from the cube list and press Next (See Figure 19).



Data Connection Wizard

Select Database and Table

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

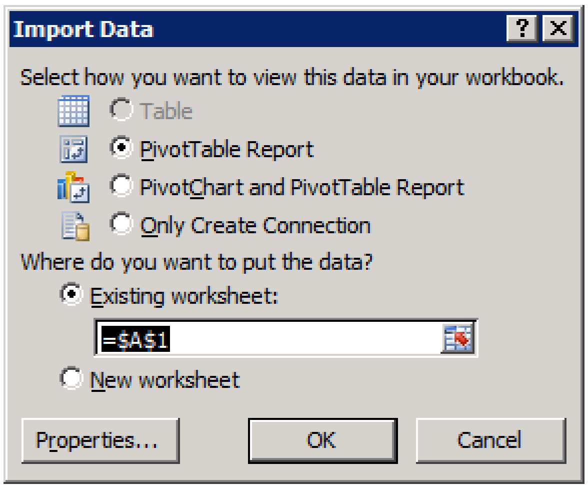
Select the database that contains the data you want: SSAS Tutorial

[✓] Connect to a specific cube or table:

Name	Description	Modified	Created	Type
SSAS Tutorial		7/19/2012 6:10:04 PM		CUBE

Figure 19 Data Connection Wizard

- Click Finish in the next window.
- An 'Import Date' window will appear. Choose 'PivotTable Report' option and select 'Existing worksheet' option, and write =$A$1 in the textbox (It should already be written automatically), and press OK (See Figure 20).


Import Data

Select how you want to view this data in your workbook.

( ) Table
[✓ PivotTable Report]
( ) PivotChart and PivotTable Report
( ) Only Create Connection

Where do you want to put the data?

[✓ Existing worksheet:] =$A$1
( ) New worksheet

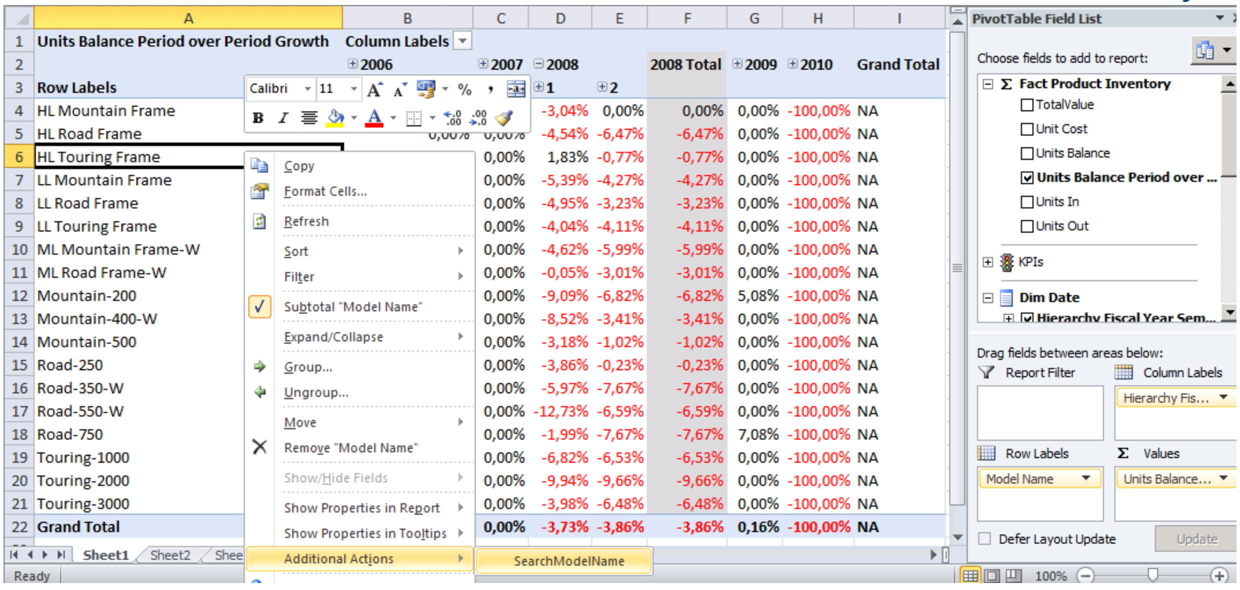
Figure 20 Import Data window

The PivotTable Tools will be configured, so you can analyze the data using various options in PivotTable.

- Select 'Units Balance Period over Period Growth' option from the 'Fact Product Inventory'.
- Select 'Hierarchy Fiscal Year Semester Quarter' from the 'Dim Date' Dimension.
- Expand 'Dim Product' and open 'More Fields' and select 'Model Name'.
- You should have 'Units Balance Period over Period Growth' as 'Column Lables', 'Hierarchy Fiscal Year Semester Quarter' as Values and 'Model Name' as 'Row Lables'.

Your screen should look like Figure 21. If you right click on one of 'Model Name', you will see that there is an item called SearchModelName under the 'Additional Actions'. This item is the action that you defined in Actions section. It enables you to search for this model name in Google. Moreover, some of the values are red, which is the expression that you wrote when defining the calculated member.

MS Excel provides various facilities to analyze OLAP data, so you can spend time to get familiar with it and understand how you could analyze the multi-dimensional data.



Units Balance Period over Period Growth	Column Labels							
	2006	2007	2008		2008 Total	2009	2010	Grand Total
Row Labels			1	2				
HL Mountain Frame			-3,04%	0,00%	0,00%	0,00%	-100,00%	NA
HL Road Frame	0,00%	0,00%	-4,54%	-6,47%	-6,47%	0,00%	-100,00%	NA
HL Touring Frame		0,00%	1,83%	-0,77%	-0,77%	0,00%	-100,00%	NA
LL Mountain Frame		0,00%	-5,39%	-4,27%	-4,27%	0,00%	-100,00%	NA
LL Road Frame		0,00%	-4,95%	-3,23%	-3,23%	0,00%	-100,00%	NA
LL Touring Frame		0,00%	-4,04%	-4,11%	-4,11%	0,00%	-100,00%	NA
ML Mountain Frame-W		0,00%	-4,62%	-5,99%	-5,99%	0,00%	-100,00%	NA
ML Road Frame-W		0,00%	-0,05%	-3,01%	-3,01%	0,00%	-100,00%	NA
Mountain-200		0,00%	-9,09%	-6,82%	-6,82%	5,08%	-100,00%	NA
Mountain-400-W		0,00%	-8,52%	-3,41%	-3,41%	0,00%	-100,00%	NA
Mountain-500		0,00%	-3,18%	-1,02%	-1,02%	0,00%	-100,00%	NA
Road-250		0,00%	-3,86%	-0,23%	-0,23%	0,00%	-100,00%	NA
Road-350-W		0,00%	-5,97%	-7,67%	-7,67%	0,00%	-100,00%	NA
Road-550-W		0,00%	-12,73%	-6,59%	-6,59%	0,00%	-100,00%	NA
Road-750		0,00%	-1,99%	-7,67%	-7,67%	7,08%	-100,00%	NA
Touring-1000		0,00%	-6,82%	-6,53%	-6,53%	0,00%	-100,00%	NA
Touring-2000		0,00%	-9,94%	-9,66%	-9,66%	0,00%	-100,00%	NA
Touring-3000		0,00%	-3,98%	-6,48%	-6,48%	0,00%	-100,00%	NA
Grand Total		0,00%	-3,73%	-3,86%	-3,86%	0,16%	-100,00%	NA

Figure 21 Excel PivotTable toolset

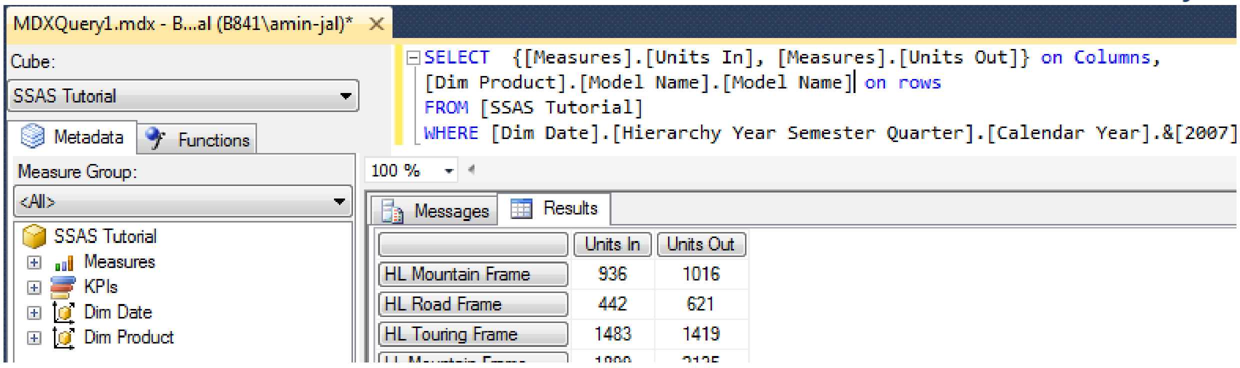
SQL SERVER MANAGEMENT STUDIO (SSMS)

SQL Server Management Studio (SSMS) helps us to manage the OLAP databases, which were deployed to the server. To run the SSMS, choose 'SQL Server Management Studio' from Start → All Programs → Microsoft SQL Server 2012 → SQL Server Management Studio. When you run it, you will be prompted to choose the appropriate server type. Select 'Analysis Services' as server type and choose appropriate server name from the combo box. Then, click 'Connect'.

Open the 'Databases' folder from the OLAP server. The 'OLAPTutorial' OLAP database should be appeared. If you expand it, you will see the structure of your OLAP as you designed and deployed in in BIDS. Spend time to consider different parts of your OLAP database. In the next section, you will see how you can query from the data in SSMS.

MDX

Multidimensional Expressions (MDX) is a query language with which you can query from SSAS. In order to write a query, right click on the 'OLAPTutorial' in SSMS and select 'New Query → MDX'. SSMS opens a window in which you can write your script (see Figure 22).



```
SELECT  {[Measures].[Units In], [Measures].[Units Out]} on Columns,
[Dim Product].[Model Name].[Model Name] on rows
FROM [SSAS Tutorial]
WHERE [Dim Date].[Hierarchy Year Semester Quarter].[Calendar Year].&[2007]
```

	Units In	Units Out
HL Mountain Frame	936	1016
HL Road Frame	442	621
HL Touring Frame	1483	1419

Figure 22 SSMS MDX Editor

The cube combobox shows the OLAP database for which you will write a MDX. Choose 'SSAS Tutorial' from the list. You can write you MDX here and execute them by pressing F5, or clicking on the Execute button. We will consider three examples as following:

EXAMPLE 1

Imagine that we need to know the 'Units In' and 'Units out' of each Product Model Names in 2007. To retrieve such information, we can write the following expression:

```
SELECT  {[Measures].[Units In], [Measures].[Units Out]} on Columns,
[Dim Product].[Model Name].[Model Name] on rows
FROM [SSAS Tutorial]
WHERE [Dim Date].[Hierarchy Year Semester Quarter].[Calendar Year].&[2007]
```

Each MDX consists of at least two parts, i.e. 'Select' and 'From'. The 'Select' part of MDX specifies the data that you are going to fetch from the OLAP. The 'From' part of the MDX specifies the OLAP structure from which the data should be selected. The 'Where' part is optional, and it limits the select statement to retrieve specific information. In this example, we select the *[Units In]* and [Units out] on columns. We also select the [Model Name] on rows. To limit our information to a specific year, we wrote the where clause. This MDX returns the 'Units In' and 'Units Out' for each product model name in 2007.

EXAMPLE 2

We defined a calculate member in Figure 16 to calculate the Units Balance Period over Period Growth. Imagine that we need to know this percentage for the fiscal year 2008 for different class of products. To do that, write the following expression in MDX editor:

```
SELECT
       [Dim Product].[Class].[All].Children On 0
FROM [SSAS Tutorial]
WHERE
       (
             [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal
Year].&[2008],
             [Measures].[Units Balance Period over Period Growth]
       )
```

This expression, select different product classes from the cube and limit them to fiscal year 2008. It was very straightforward expression, but what if we have not defined such calculate member when defining our OLAP? In the next example we will see how we could define calculated members in MDX.

EXAMPLE 3

To define a calculated member in an MDX we could use the following expression:

```
With Member [MemberName] AS ...
Select ...
From ...
```

We have defined the expression in Figure 16, so we copy that expression and make a new one using previous MDX. The result will be like the following MDX:

```
WITH MEMBER [Measures].[UBPG] AS
Case
// Test for current coordinate being on (All) member.
When [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember.Level Is
     [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[(All)]

Then "NA"
// Test to avoid division by zero.
When IsEmpty
     (
       (
        ParallelPeriod
        (
           [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year],
           1,
                 [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember
        ),
       [Measures].[Units Balance]
       )
     )
Then Null

Else (
       ( [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember,
           [Measures].[Units Balance] )
         -
       (
        ParallelPeriod
        (
           [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year],
           1,
              [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember
        ),
        [Measures].[Units Balance]
       )
        )
        /
     (
       ParallelPeriod
       (
          [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal Year],
          1,
             [Dim Date].[Hierarchy Fiscal Year Semester Quarter].CurrentMember
       ),
       [Measures].[Units Balance]
     )
End
SELECT
       [Dim Product].[Class].[All].Children On 0
FROM [SSAS Tutorial]
```

```
WHERE
       (
              [Dim Date].[Hierarchy Fiscal Year Semester Quarter].[Fiscal
Year].&[2008],
              [Measures].[UBPG]
       )
```

We changed the name from [Measures].[Units Balance Period over Period Growth] to [Measures].[UBPG] in order to prevent confusion. This expression defines a new calculated member called [Measures].[UBPG] and uses it in the select statement.

You will learn about the MDX in OLAP lecture more.

CONCLUSION

This tutorial is designed to give you very basic understanding of how to work with an OLAP Server. It shows how you can make a new project and design your OLAP structure. It also leaded you to deploy your project to the OLAP server and show how you can query it.

BIBLIOGRAPHY

1. **Kimball, Ralph, et al., et al.** *The Kimball Group Reader Relentlessly Practical Tools for Data Warehousing and Business Intelligence.* s.l. : Wiley Publishing, Inc., 2010. 978-0-470-56310-6.

2. **Melomed, Edward, et al., et al.** *Microsoft SQL Server 2005 Analysis Services.* s.l. : Sams Publishing, 2007. 0-672-32782-1.