Notes for completing assignments:

In this assignment, you are going to build an OLAP cube for the first time. Please provide screenshots for the steps and add appropriate explanatory notes using the information provided in the lectures.

It is important to see how this assignment closely mirrors the activities on the job. It is required that you research the benefits that a data warehouse brings to a business. This would be reflected in the objectives. Your conclusions would reflect your experience in building the cube.

Please download and attach the AdventureworksDW2012\_data.mdf file to your database.

In the evaluation of this assignment you must have a working cube accompanied by a report.

You will be evaluated on the following aspects of a working cube and the report.

Evaluation scheme:

|  |  |
| --- | --- |
| Dataview | 5 |
| Fact Table | 5 |
| Dimensions | 5 |
| Hierarchies | 5 |
| Process Cube | 5 |
| Aggregates | 5 |
| Objectives | 10 |
| Conclusions | 10 |
| Overall Report | 10 |
| Total | 60 |

Your report should have at least the following sections:

Objectives:

Here you will list the anticipated benefits of the cube. As you go through the exercise you will constantly revise this.

Based on your research, document about four aspects of building a data warehouse, the advantages to a business, the resources it consumes etc.

Each “time for action” section:

Summarize the steps and provide the screen shots of the intermediate processes and summarize the results.

Conclusion:

Provide screen shots of the final results, the aggregation of various hierarchies.

Provide an overall assessment of the process of building a cube.

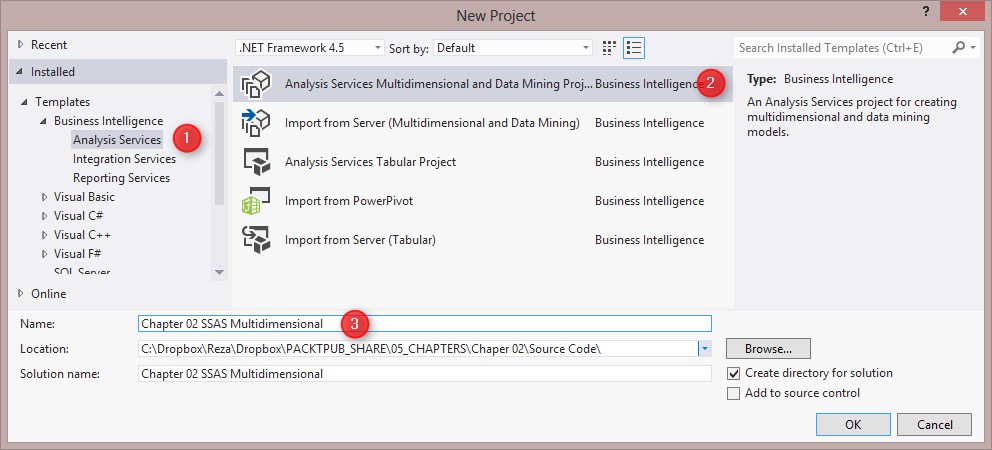
# Developing your first cube

## Time for action – creating an Analysis Services project

In this example and other examples of this book, we will use SSDT for Visual Studio 2012. You can download and install it from **http://www.microsoft.com/en-us/download/ details.aspx?id=36843**.

After installation, perform the following steps:

1. Open SQL Server Data Tools by clicking on **Start**, then navigate to your Microsoft SQL Server 2012 folder and under that, choose **SQL Server Data Tools for Visual Studio 2012**.
2. Go to the **File** menu and under **New**, select **Project**.
3. In the **New Project** window, under templates in the left-hand side pane, click on **Analysis Services** and from the list of templates in the main pane, choose **Analysis Services Multidimensional and Data Mining Project**. Name the project **Chapter 02 SSAS Multidimensional**, as shown in the following screenshot:



What just happened?

In this example, we used SQL Server Data Tools to create our first Analysis Services project.

After creating the project, you will see an empty project with folders listed as follows:

|  |  |
| --- | --- |
| **Folder name** | **Description** |
| **Data Sources** | This contains source connections to databases |
| **Data Source**  **Views** | This contains diagrams of tables and views with relationship and metadata information |
| **Cubes** | The OLAP cube |
| **Dimensions** | The SSAS database dimensions |
| **Mining**  **Structures** | Data Mining structures (refer to *Chapter 7*, *Data Mining – Descriptive*  *Models in SSAS* and *Chapter 8*, *Identifying Data Patterns – Predictive Models in SSAS* for more information) |
| **Roles** | Security roles and permissions |
| **Miscellaneous** | Any other file that is related to this project |

## Time for action – creating the first cube

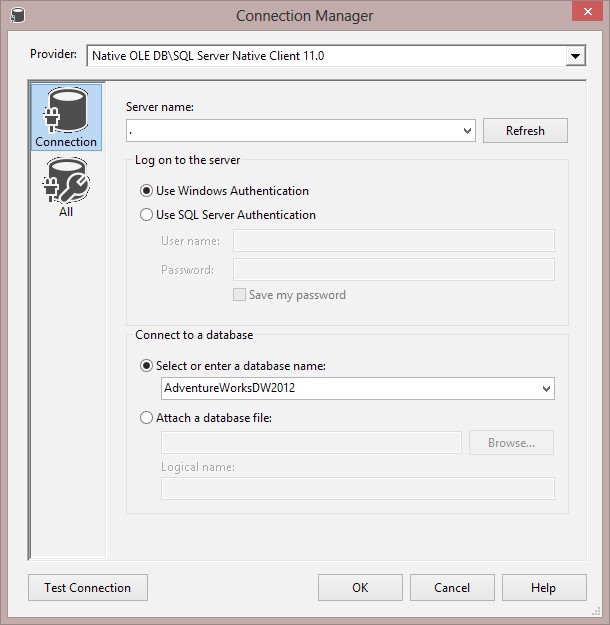
There are two ways to create a cube: through **Cube Wizard** or by creating an empty cube. When we create a cube from the wizard, the relationships between measures and dimensions will be created automatically based on the underlying **Data Source View** (**DSV**) after choosing it. However, if we create a cube from an empty one, we need to set up a relationship between the measure groups and dimensions in the cube designer. In the next example, we create the cube using Cube Wizard.

1. The first step is to create a data source connection. To create a data source connection, right-click on a data sources folder.
2. Open the SSAS project and select **New Data Source**. As shown in the next screenshot, create a connection to the **AdventureWorksDW2012** database. In the **Impersonation Information** field, enter the username and password of a domain/ windows account that has the appropriate permissions for that database. Name the

data source

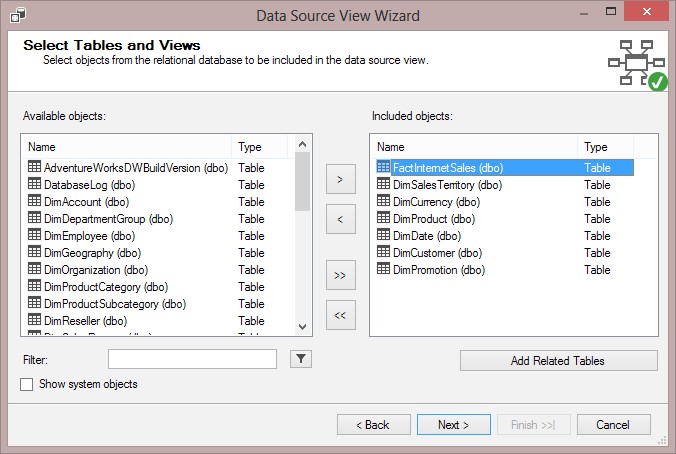
**AdventureWorksDW2012**

.



1. There is one more step before creating the cube, and that step is the creation of the Data Source View. Right-click on the **Data Source Views** folder and choose **New**

**Data Source View**. Choose the data source from step 1. In the **Select Tables and View** step, choose the tables as shown in the next screenshot and name the data source view **AdventureWorksDW2012**:



You

can also use the

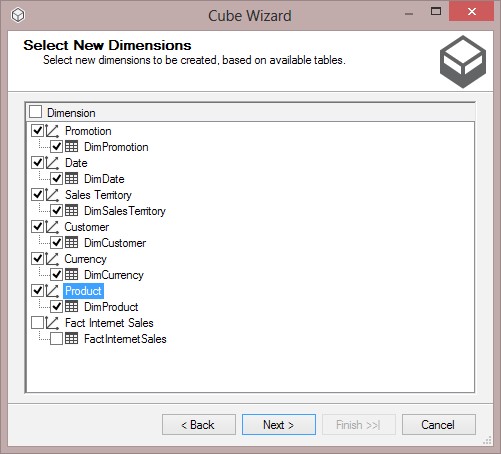
**Add Related Tables**

button, as

shown in the previous

screenshot, to include all objects related to the selected object.

1. After creating DSV, you will see a database style diagram of tables with their relationship in the design area. In this example, we just use the DSV as is, but in the next section, we will show you how to modify the DSV.
2. Now, it is time to create the cube. Right-click on **Cubes** and select **New** **Cube**. Follow the steps in the wizard, and in the **Select Creation Method** step, choose **Use Existing Tables**.
3. In the **Select Measure Group Tables** step, select the checkbox of the **FactInternetSales** table.
4. In the **Select Measures** step, just leave all the measures as checked. Rename the measure group **Internet Sales**.
5. In the **Select New Dimensions** step, leave all the tables as checked except for **Fact Internet Sales**. Rename all the options in **Dimension** in this step, and remove **Dim** from their name, as shown in the following screenshot:



***9***

***.***

Rename the cube

**Internet Sales**

.

What just happened?

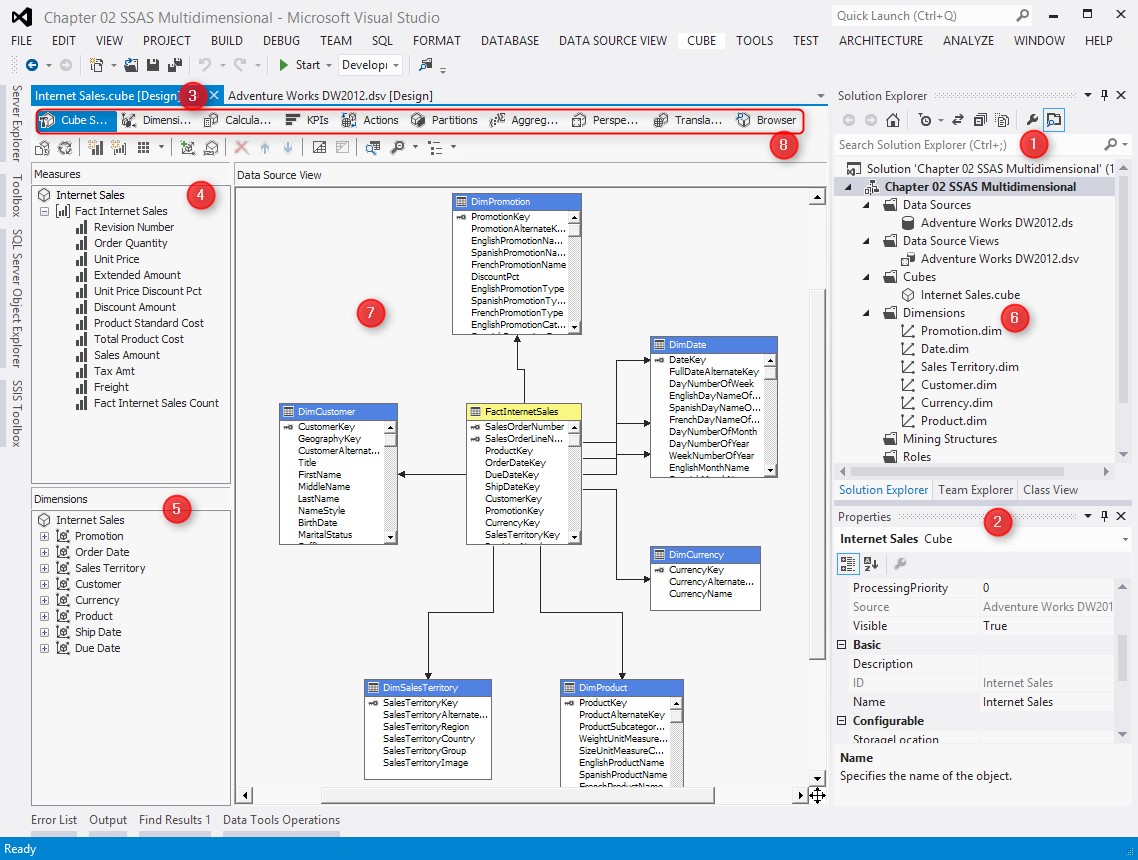
In the first step of your example, you created a data source connection to a data warehouse. The credential used to create this connection is important because SSAS will use that credential to connect to the underlying database.

In the next step, a DSV is created from the data source. You can add as many tables and views as you want to the data source view; the data source view creates the base structure for cube development.

In steps 4, 5, and 6, you created a cube through **Cube Wizard** from the existing tables in the data source view. To create cubes, you should define your measure groups and dimensions. Measure groups are similar to the Fact tables, each measure group may contain one or more facts. In this example, the **FactInternetSales** table is selected as a measure group, and it contains measures such as **Sales Amount** and **Order** **Quantity**.

In step 7, we saw that dimensions are the second important items that should be defined when you create the cube. In the example, dimensions such as **Date**, **Product**, and **Sales Territory** are used. Each dimension may be connected to a dimension table in DSV or to a view that is a result of joining multiple tables of the database.

After creating the cube, you will see a view of the cube in the cube designer. The following screenshot shows the different parts of SSDT:



The **Solution** **Explorer** pane (number **1** in the screenshot) is part of SSDT that shows projects and files, data sources, DSVs, cubes, and other project-related items.

The **Properties** window (number **2** in the screenshot) is the place where you can view and change the properties of the selected object; for example, in the previous screenshot, the **Properties** window shows the properties of the **Internet Sales** cube.

The cube designer (number **3** in the screenshot) is the main designer in the SSDT in which you will spend most of your time as an SSAS developer. Here, you can view the layout of the cube's measure group and dimensions (number **7** in the screenshot); you can go to different tabs (number **8** in the screenshot), such as **Dimension** **Usage**, **Partitions**, **Browser**, and **KPIs**. We will walk through many of these tabs in this chapter.

A list of the **Measure** groups and their measures (number **4** in the screenshot) can be viewed in the **Cube Structure** tab. You can also see two different lists of dimensions: first is the database's dimensions (number **6** in the screenshot) and second is the cube's dimensions (number **5** in the screenshot). The difference between the cube's and the database's dimension is that there might be some dimensions in the database that are not used in this cube (for example, consider an SSAS project with multiple cubes). On the other hand, there might be a single database dimension that is used multiple times in one cube, which we will call the role-playing dimension (for example, there is only one database dimension, **Date**, but there are three role-playing dimensions in the cube, which are named **Order** **Date**, **Ship** **Date**, and **Due** **Date**).

One of the benefits of creating a cube from a wizard is that it will automatically create all the relationships between the database dimensions and the measure groups, based on the relationship defined in the DSV.

## Time for action – viewing the cube in the browser

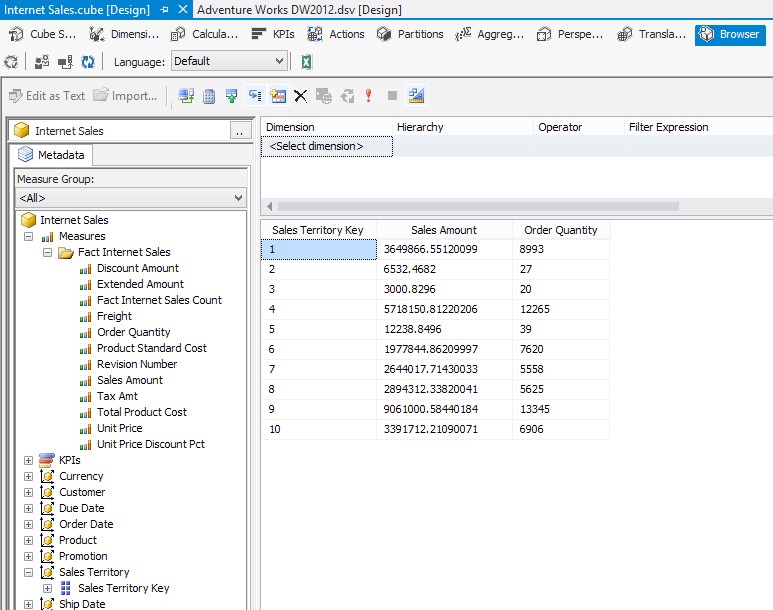
After creating the cube, you can view it in the cube browser. There are two ways to open the cube browser: through SQL Server Management Studio and through the built-in cube browser in the SSDT. The SSMS cube browser will be useful when we deploy the cube to an SSAS Server (for details on deployment, refer to the *Deploying and processing* section of this chapter). For this example, we use the SSDT cube browser.

To browse the cube, processing is required. Processing is the process of extracting data from a data warehouse, applying the cube's calculations and structures to it, and loading it into the OLAP cube's structure. Perform the following steps to view the cube:

***1.*** In **Solution Explorer**, right-click on the project **Chapter 02 SSAS Multidimensional** and choose **Process**. ***2.*** You will see a dialog box stating **The Server contents appears to be out of date.**

**Would you like to build and deploy the project first?**. Answer it by clicking on **Yes**.

1. In the **Process** window, leave all the configurations as the default and click on **Run**.
2. After completing the **Process Progress** step, close the **Process** window.
3. Go to the **Browser** tab, which is the last tab in the cube designer.
4. If you've got a message saying that the database does not exist or the cube is not up to date, try to reconnect the browser. You can click on the **Reconnect** button, which is the third icon from the top left-hand side of the **Browser** tab.
5. In the **Browser** tab, from the **Metadata** pane, under the **Internet Sales** measures, drag-and-drop **Sales Amount** into the main empty pane in the middle. Do the same for the **Order Quantity** measure.
6. You will see a grand total of **Sales Amount** and **Order Quantity** calculated from the cube.
7. From the **Metadata** pane, under the **Sales** **Territory** dimension, drag-and-drop **Sales** **Territory** **Key** into the main pane as shown in the following screenshot:



You deployed and processed the cube in the previous example in order to be able to view the cube in the browser. In the last section of this chapter, we will discuss deployment and processing in more detail.

Viewing the cube in the browser of SSDT helps SSAS developers to check the frontend view of the cube from the user's perspective. In this example, you've seen the grand total of **Sales Amount** shown easily with a drag-and-drop in the browser window.

Slicing and dicing the **Sales** **Amount** by **Sales** **Territory** is also easily done by dragging-anddropping the **Sales** **Territory Key** dimension in the browser. However, you probably noticed that the **Sales** **Territory** dimension only has one attribute, which is a key, and the key numbers don't make sense for a business user. So, for the next section, we need to bring descriptive information as additional attributes in the **Sales** **Territory** dimension and other dimensions of the cube.

On the other hand, the **Sales** **Amount** and **Order Quantity** dimension scan be shown with thousand separators and with a suitable format (for example, currency for the sales amount) in order to be more appropriate from the user's point of view. In the next section, we will change the display format of measures.

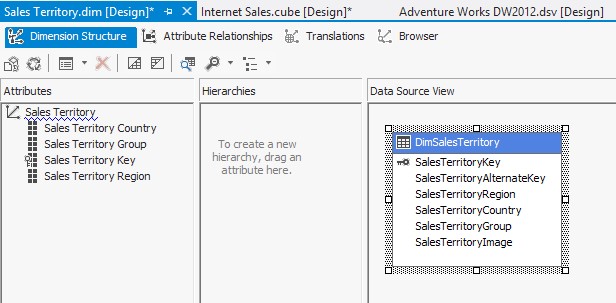
# Dimensions and measures

At the time of cube development in the SSAS, there might be many times when you would need to make changes to the dimensions and measures. For example, you might need to change the display format of a measure or change the aggregation function of it. For dimensions, you might also need to add/remove an attribute or change the relationship between the attributes. In this section, we will add some attributes to the **Sales** **Territory** dimension and also make changes in the display format of the measures.

## Time for action – using the Dimension Designer

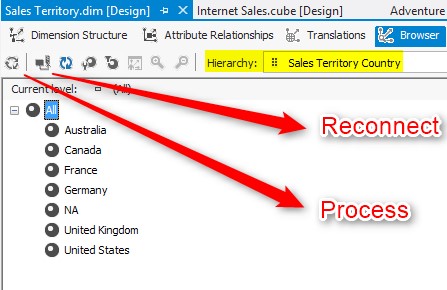
The **Dimension Usage** tab is a specific designer for dimensions in the SSDT. In the next example, you will see how to create new attributes using the Dimension Designer:

1. In the **Solution Explorer**, right under dimensions, double-click on **Sales Territory**.
2. A new designer window will be opened; this designer is called a Dimension Designer.
3. In the **Dimension Structure** tab, drag-and-drop **Sales Territory Region** from the **Data** **Source** **View** pane to the **Attributes** pane.
4. Do this for **Sales Territory Group** and **Sales** **Territory** **Country** as well.
5. Select **Sales Territory Key** in the **Attributes** pane, and then in the **Properties** window, change the **AttributeHierarchyVisible** property of this attribute to **false**.



1. Save the changes and process the dimension. You can process the dimension with the **Process** button, which is the second button from the top left-hand side of the dimension structure tab.
2. Go to the **Browser** tab in the Dimension Designer and then reconnect to a process

(you can see how to reconnect to a process in the next screenshot). Select **Sales** **Territory** **Country** in the hierarchy drop-down list and you will then see the dimension member values listed in the browser:



The Dimension Designer used in this example adds new attributes to the **Sales Territory** dimension. To create a new attribute, you can simply drag-and-drop it from the data source view, but sometimes the data source view doesn't contain the appropriate column, which forces you to change the data source view (in the next chapter, you will see how to make changes in the DSV).

You can also change the properties of the dimension attributes in the **Dimension** **Structure** tab. For example, we changed **AttributeHierarchyVisible** of **Sales Territory Key** to false, and as a result, this attribute was not shown directly in the browser. There are some properties and configurations for each attribute. The following list shows the most useful properties of attributes:

|  |  |
| --- | --- |
| **Property** | **Description** |
| **AttributeHierarchyDisplayFolder** | You can specify a folder name, and the attribute will be listed under that folder name under **Dimension** in the browser. |
| **AttributeHierarchyEnabled** | As a default function, each attribute creates a hierarchy. Which hierarchy will create separate aggregations is based on each member value. This feature is very useful when you apply it on attributes that have a number of unique values, which is much lower than the number of all the dimension members' values (such as city), but it will be better to disable it for attributes such as the address line that might be different on each record of the dimension. |
| **AttributeHierarchyVisible** | Controls the visibility of the attribute in the browser for the client tools, but this attribute is still accessible from MDX queries. |
| **DiscritizationBucketCount** | Number of buckets when you want to discretise a continuous attribute. |
| **DiscritizationMethod** | Used to choose a discretization method. |
| **IsAggregatable** | Defines whether the given attribute is aggregatable or not. |
| **OrderBy** | Defines how to place the attributes in hierarchy according to the order. |
| **OrderBy Attribute** | Defines the related attributes for which the type of ordering will be executed.. |
| **NameColumn** | This column acts as a label for each member. |
| **KeyColumns** | This column(s) acts as a key for each member. |

To run this example, you need to deploy the changes and the process again; this step is required after each change that you make in the metadata of the SSAS cube, dimension, or measures. For the next example, we will use the **OrderBy** and **OrderBy Attribute** properties to change the ordering of a dimension attribute.

## Time for action – change the order of the Month attribute

For this example, we will add some attributes to the date dimension, such as a month attribute, and we will then change the ordering of that attribute to show the correct order as shown in the following steps:

1. Double-click on the **Date** dimension in **Solution** **Explorer**.
2. In the **Dimension** designer, go to the **Browser** tab; you can see all the members of the **Date Key** attribute there. Members of the **Date** **Key** attribute are integer values with the YYYYMMDD format. This format is not well formed from the business user's perspective. In this example, we will change this format and also add two other attributes for the year and month.
3. Go to the **Dimension** **Structure** tab, select **Date Key** in the **Attributes** pane, then in the **Properties** window, click on the ellipsis button of the **NameColumn** property. In the **Name Column** dialog box, select **FullDateAlternateKey** and click on **OK**.
4. Then rename the **Date Key** attribute to **Date**.
5. Drag-and-drop **CalendarYear** from the **Data** **Source** **View** pane to the **Attributes** pane.
6. Drag-and-drop **MonthNumberOfYear** from the **Data** **Source** **View** pane to the **Attributes** pane.
7. As the **Month Number of Year** view only shows values such as **1**, **2**,… **12**, we need to change it to show the month names. So, change the **NameColumn** property of this attribute to **EnglishMonthName**. Also, rename this attribute to **Month**.
8. Process the dimension and go to the **Browser** tab; if you choose **Date** in the hierarchy drop-down list, you will see values such as **2005-01-01**, which is what we want.
9. If you choose **Month** in the hierarchy drop-down list, you will see that the month names are visible, but they are not in the correct order.
10. Go back to the **Dimension** **Structure** tab. Select the **Month** attribute, and then in the **Properties** window, change the **OrderBy** property to **Key**.
11. Process the dimension and go to **Browser**, and you will see that the month's values are shown in the correct order this time.

In this example, you used two other properties of the dimension attributes. You used those properties to change the order of the values and also to change the display values of an attribute. You also learned how to make changes in the dimension structure and check results in the browser.

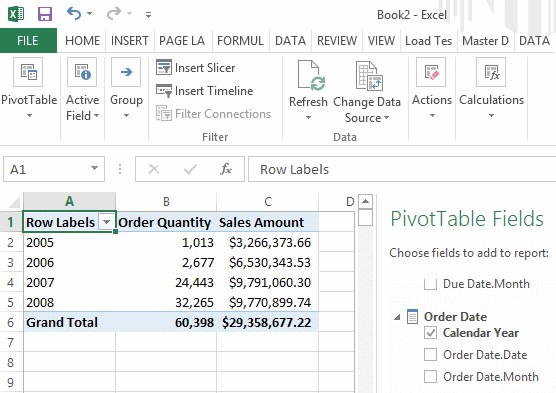
In step 3 and step 7, we changed the label value of the attributes with the help of **NameColumn**. While configuring the **NameColumn** property to another column in the data source view, the value of that column will be shown in **Browser**, but the value of the **KeyColumn**(s) will be used in aggregations and calculations. You did that for **Month** and **Date** full format in the preceding example.

Using the **OrderBy** attribute, you changed the order of the **Month** attribute to **Key**. This means that this attribute will be sorted based on **KeyColumn**, and because the **KeyColumn** is **MonthNumberOfYear** and it is an integer value, it will be ordered correctly (refer to step 10). It is also possible to sort one attribute based on another attribute's **Name** or **Key** column (using the **OrderByAttribute** property).

## Time for action – modifying the measure properties

After modifying the dimension attribute's properties, it is time to change the measure's properties. Properties such as display format, display folder, and aggregation function can be changed in the **Properties** window. Perform the following steps:

1. In SSDT, go to the **Cube** designer, and in the **Cube structure** tab, select the **Sales** **Amount** measure. Change the **FormatString** property of this measure to **Currency**.
2. Select the **Order** **Quantity** measure and change **FormatString** of this measure to **#,##**.
3. Go to the Excel browser; there is an option to view the cube in Excel, and you need to click on **Analyze** in the Excel icon in the browser.
4. In the Excel browser, choose **Sales** **Amount** and **Order** **Quantity**, and from **Order** **Date**, choose **Calendar** **Year** as shown in the following screenshot:



What just happened?

You changed the display format of the measures by configuring the **FormatString** property.

You can also change the **AggregationFunction** to **Sum**, **Average**, **Count**, and so on. Also, in this example, you saw a new browser for the Analysis Service cube.

Excel is a great tool for business users; it can connect to the SSAS cube and shows measures and dimension attributes as formatted. Business users are quite familiar with Excel because it can be easily found on every system that has Microsoft Office installed. Also, Excel has many features for filtering and charting, which makes it more comfortable for users to work with data.

# Data Source View

In the previous sections, we explained what the **Data Source View** (**DSV**) is and how to create DSV. However, sometimes, there is a requirement to change the DSV. For example, you might want to add a calculated column in the DSV with SQL commands and functions, you might want to create a relationship between two views, or you might need to write a query and create a DSV table based on that query. All of these modifications can be done in the DSV, and in this section, we will go through some of them.

## Time for action – creating a Named Calculation

In this exercise, we will create a new calculated column in the DSV for the full name because the customer dimension only contains the first name and last name separately. You will see how we can use T-SQL commands to create a Named Calculation. Perform the following steps to create a Named Calculation.

1. Go to the **Adventure Works DW2012 DSV** designer.
2. In the **Customer** dimension, you can see that there are three name columns as **First name**, **Middle name**, and **Last name**, but there is no full name. We want to have the full name shown as the main customer attribute of the customer dimension in the cube. So, we will create a column in DSV for that.
3. Right-click on **DimCustomer** and select **New** **Named** **Calculation**.
4. Set the column name as **Full** **Name**, and write the following code in the **expression** area:

**FirstName+' '+ISNULL(MiddleName+' ','')+LastName**

1. Click on **OK** and you will see the new column added to the table. Right-click on **DimCustomer** and select **Explore** **Data**. Some sample data rows from the **Customer** table will be shown in another window and you can also see the **Full** **Name** column populated at the end of the column's list as well.
2. Go to the **Customer** Dimension Designer, select the **Customer** **Key** attribute, and change the **NameColumn** to **Full** **Name** from **DimCustomer**. Also, rename the **Customer** **key** to **Customer**.
3. Browse the **Customer** dimension (after deploying and processing), and you will see that the full name of the customer appears as **Customer** **Values** in the browser.

What just happened?

Named Calculations are very useful when you want to add calculated columns to the underlying database, but for some reason, you cannot do that in the database. For example, your user account only has read-only permissions on the database. In such cases, you can benefit by creating Named Calculations.

As you've seen in this example, Named Calculations are created based on the **expression** area. The language of the expression is based on the underlying database. This means that if the underlying database is an Oracle database, you can use functions such as **rownum** or **decode**, which work in an Oracle environment. If the underlying database is SQL Server, you can use functions and T-SQL codes that are acceptable in that SQL Server version.

You also saw how to check the result of the execution of Named Calculations with the **Explore** **Data** option. You can also change the sampling settings for the explored data.

## Time for action – using a Named Query

Sometimes, you need to write queries instead of bringing tables and views as is from the database (assume that we don't have access to the underlying database in order to create views or change the structure of tables there). Named Query is a query that runs on the database and its result will be shown as a table in DSV. In this example, we will use a Named Query to bring the product category and subcategory along with the product table columns, by performing the following steps:

1. In the **Data** **Source** **View** designer, right-click on the **DimProduct** table and under **Replace** **Table**, choose **With** **New** **Named** **Query**.
2. In the **Create** **Named** **Query** window, add the **ProductCategory** and

**ProductSubCategory** tables, and the designer will write joined statements itself.

1. After that step, choose **EnglishProductCategoryName** and **EnglishProductSubcategoryName** from the columns list.
2. Explore data of the **DimProduct** Named Query and review the result.

What just happened?

In this example, we used a Named Query and a query designer to write customized queries on the database. The query language used is T-SQL because the underlying database is SQL Server. However, it can be different based on the database.

You can also create logical keys for Named Queries or views. You can also create relationships between tables, views, and Named Queries in the DSV. These features in the DSV will provide the ability to develop a powerful base for the cube.

# Using dimensions

The junction between dimensions and measure groups is defined in the dimension usage area. There are multiple types of relationships between a measure group and dimension. In this section, we will go through the dimension usage and create a new connection for a business requirement.

## Time for action – adding a Fact relationship

As a business requirement, you need to add the order line number as a dimension attribute and show it in the browser. As you know, in the current cube, we have an **Internet** **Sales** measure group, which has some measures such as order quantity and sales amount, but we need to fetch the order line number from **FactInternetSales** as a dimension attribute (this is degenerate dimension, which is explained clearly in *Chapter 1*, *Data Warehouse Design*).

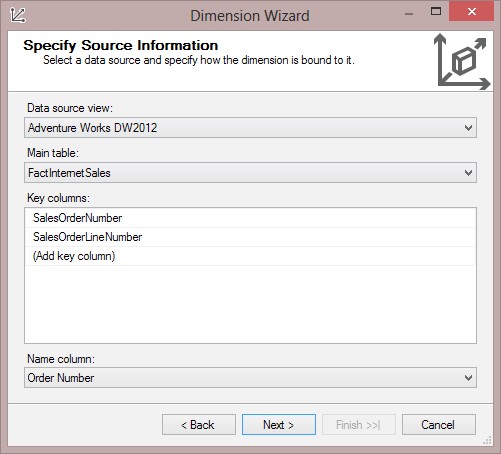
Perform the following steps to add a Fact relationship:

1. Open the **Data** **Source** **View** designer and explore data in the **FactInternetSales** table. You will see that there is a **SalesOrderNumber** and **SalesOrderLineNumber** column there. In this exercise, we will create a dimension based on these columns.
2. Add a Named Calculation in the **FactInternetSales** table and name it **Order** **Number**. Write the following expression to calculate the concatenation of the order line and order number:

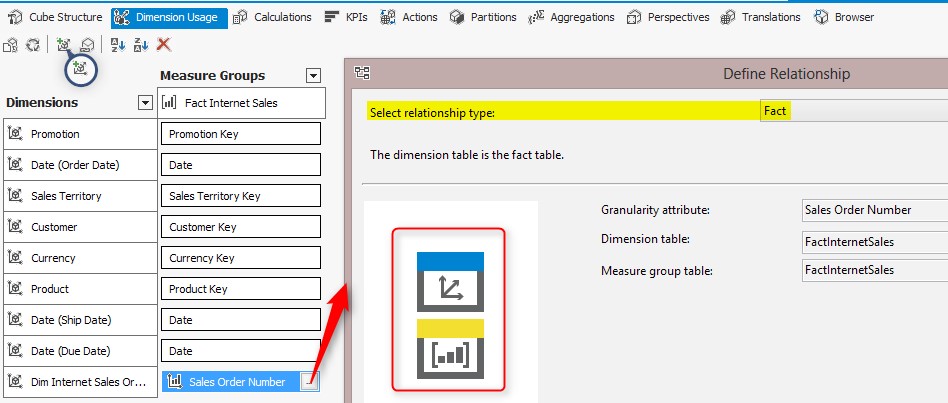
**'Order Number: '+SalesOrderNumber+', Order Line Number:**

**'+convert(varchar(max),SalesOrderLineNumber)**

1. Explore data in the **FactInternetSales** table. You will see that **Order** **Number** creates a concatenation of two order line columns such as **Order Number: SO43697** and **Order Line Number: 1**.
2. In the **Solution** **Explorer** option, right-click on the **dimensions** folder and select **New** **Dimension**.
3. In the **Select** **Creation** **Method** step, choose **use an existing table**.
4. Choose **FactInternetSales** as the table, leave the **SalesOrderNumber** and **SalesOrderLineNumber** columns in the key column area, and choose **Order Number** as **Name column**, as shown in the following screenshot:



1. In the **Select** **Related** **Tables** step, uncheck all the tables.
2. In the **Select** **Dimension** **Attributes** step, uncheck all the attributes and only check the **Sales Order Number** option.
3. In the last step, rename the dimension to **Dim** **Internet** **Sales** **Order**.
4. The new dimension will be created under the **dimensions** folder in **Solution** **Explorer**. Go to the cube designer; in the **Cube** **Structure** menu, you will see that the new dimension does not exist there. We will add this dimension to the cube in the next steps.
5. Go to the **Dimension** **Usage** tab in the cube designer (the second tab). Click on the third icon on the top left-hand side, which is **Add** **Cube** **Dimension**.
6. In the **Add** **Cube** **Dimension** dialog box, choose **Dim** **Internet** **Sales** **Order**.
7. In the matrix, you will see that this dimension is related to the **Internet** **Sales** measures group based on the **Sales Order Number**. When you double-click on the **Sales Order Number** measure group, you will see that the relationship type is already set as **Fact**, as shown in the following screenshot:



1. After deploying and processing the dimension, check the result in the cube **Browser**. You can view **Sales** **Order** **Number** as a dimension attribute and see **Sales** **Amount** and other measures sliced and diced by this attribute.

What just happened?

One of the most important parts of modeling in SSAS Multidimensional is setting up the correct relationship between measure groups and dimensions. **Dimension** **Usage** is a tab where we can create, modify, or remove this kind of a relationship.

In this example, we used a descriptive column in the **FactInternetSales** table, and named it **Sales** **Order** **Number** (which is a concatenation of the order number and order line number). If you read *Chapter 1*, *Data Warehouse Design*, you will know about the degenerate dimension and you know that they are in Fact table's granularity but they don't have their own separate dimension table. This is an example of a degenerate dimension.

In this example, we created a SSAS database dimension based on the **FactInternetSales** table. In step 6, we set the key columns and the name column for the same. Then, we created a relationship in the cube between the **Internet Sales** measure group and this new dimension (step 11 and 12). As the source table for both the dimension and measure group is the same, the cube designer considered this relationship as a Fact table and set the granularity of the relationship as the key column of the dimension (step 13).

There are different types of relationships. In the previous example, you saw the **Fact** relationship, which is useful when the dimension is made up of a Fact table. The next table shows information about other relationships. By choosing each relationship, you will see an image that illustrates the relationship between the measure group and dimension (look at the red surrounded area in the screenshot of step 13).

|  |  |
| --- | --- |
| **Relationship Type** | **Description** |
| None | There is no relationship between the measure group and the dimension. |
| Regular | This is the most common relationship as it shows shows the start schema relation between the fact and the dimension in a one-to-many diagram. |
| Referenced | The relation between a measure group and a dimension based on an intermediate dimension is observed. This relationship is for snowflake diagrams, which is not recommended usually. |
| Many to many | This is used when a measure group relates to a dimension based on an intermediate dimension and measure group. |
| Fact | This is used when dimension is created based on a Fact table. |
| Data mining | This type relates data mining dimension to a measure group. |

# Hierarchies

There are two kinds of hierarchies in SSAS Multidimensional: user hierarchies and attribute hierarchies. Each attribute creates a single-level hierarchy that can be used in the designer/ browser with a combination of other attributes to create a multilevel hierarchy. A user hierarchy is a predefined multilevel hierarchy that provide easier access for the end user in the frontend. One of the most important benefits of hierarchies is the ability to drill down and drill up in many MDX client tools such as Excel and PerformancePoint.

## Time for action – creating a hierarchy

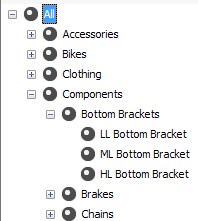
In this example, we will see how a user can create hierarchies in the browser and how a developer can create a predefined hierarchy for business requirements by performing the following steps:

1. Go to the cube **Browser**, drag-and-drop **Calendar** **Year**, **Month**, and **Date**, one after another. Then, drag-and-drop **Sales** **Amount** and **Order** **Quantity**. You will see a hierarchy created dynamically from the browser.
2. Now, go to the product's Dimension Designer, drag-and-drop the

**EnglishProductCategory** name into the **Attributes** pane, and rename it **Product**

**Category**. Do the same for **EnglishProductSubcategoryName** and rename it **Product** **Subcategory**. Rename the **Product** **Key** attribute **Product** and change its name column to **EnglishProductName**.

1. Drag-and-drop **Product** **Category** from the **Attributes** pane into the **Hierarchies** pane.
2. A new hierarchy will be created; rename it **Product** **Category** **Hierarchy**.
3. Drag-and-drop the product **Subcategory** under the **Product** **Category** option (in the **<new level>** area), and then add **Product** under that.
4. Select **Product** **Category** and change the **AttributeHierarchyVisible** option of this attribute to false. Do this for the **Product** **Subcategory** and **Product** attributes as well.
5. In the **Dimension** browser, you will see that only **Product** **Category** **Hierarchy** is listed, and the values of this hierarchy have created a tree of **Product** **Category**, **Subcategory** and **Product**.



What just happened?

In this example, you created a user hierarchy easily with just a few drag-and-drops. The hierarchy of this example is created from one table. Creating hierarchies from multiple related tables is as simple as creating a hierarchy from a single table; you just need to import attributes from the **Data** **Source** **View** pane into the attributes area and you can then create a hierarchy based on the attributes.

You also saw that after creating **Product** **Category** **Hierarchy**, we set the visibility of the attributes to false. As the user can access the **Product** **Category**, **Subcategory**, and **Product** attributes through **Product** **Category** **Hierarchy**, there is no need to have separate attributes. Making them invisible from the user's perspective will reduce the user's confusion with regards to too many attributes.

# Multidimensional eXpression, calculated members, and Named Sets

In every business, you will face situations where something new came from the business requirements that you didn't consider while designing the data warehouse. On the other hand, sometimes the requirements cannot be fulfilled in the data warehouse and should be fulfilled in the cube.

SSAS provides a way to create calculated members based on an expression language named **Multidimensional** **eXpression** (**MDX**). MDX is the query language for SSAS; all queries from the client applications are sent to the SSAS server as an MDX query, and the result of that query will be returned in a multidimensional format.

As an example for calculation of the members, assuming that you want to calculate the product of the unit price and the item quantity, you can create a calculated member for that. Also, sometimes you might require a subset of a dimension attribute, for example, a list of the top 10 customers in a specific area. The Named Set in SSAS provides the ability to create these kinds of subsets.

## Time for action – writing an MDX query

In this example, you will learn more about the MDX structure and how to write queries in the MDX language.

1. Open SQL Server Management Studio, connect to the Analysis Services instance, and click on **Chapter 02 SSAS Multi Dimensional**, and then click on **New** **Query**.
2. In the **New** **Query** window, write the following expression:

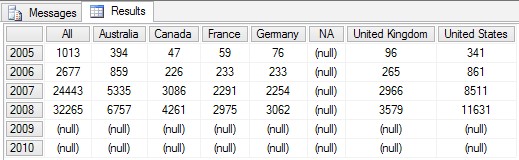
**select [Order Date].[Calendar Year].members on 0 from [Internet Sales]**

You will see a list of the years, the sales amount for each year, and the grand total. ***3.*** Now, run the following statement:

**select [Sales Territory].[Sales Territory Country].members on 0,**

**[Order Date].[Calendar].[Calendar Year].members on 1 from [Internet Sales]**

***4.*** Check the result and you will see the years listed in each row, the countries listed in each column, and the **Sales** **Amount** measure shown in each cell, which represents the sales amount of that country in that year.



What just happened?

MDX is a query language that returns the results in a multidimensional format. In this example, you saw how we define an axis in the output result set. In the first query (step 2), there is only one axis used, which is the column axis. In the second query (step 5), two axes are used, the column axis shows the countries and the row axis shows the years.

An MDX query is not limited to only two axes; you can write an MDX query with 128 axes , for example. However, the client tools are limited to show more than two axes for example, in SSMS, you cannot write a query with more than two axes because it cannot be shown as a result.

The MDX select query structure consists of the **SELECT** clause on multiple axes, **FROM** clause, and the **WHERE** clause.

**SELECT [<axis\_specification>**

**[, <axis\_specification>...]]**

**FROM [<cube\_specification>]**

**[WHERE [<slicer\_specification>]]**

There are many MDX functions to work with time, hierarchies, and other common applications that are helpful when writing MDX queries. You will work with a few of them in the following examples.

## Time for action – calculated members

Calculated members help to perform calculation based on measures, dimension hierarchies, and MDX functions. They are very helpful for covering business requirements with the help of a cube. In this example, we will create a new calculated member for **Profit**.

1. In the cube designer, go to the **Calculations** tab.
2. Create a new calculated member by clicking on the button that is magnified in the next screenshot.
3. Rename the calculated member to **[Profit]**.
4. Write the following expression in the **Expression** field:

**[Measures].[Sales Amount] - [Measures].[Total Product Cost]**

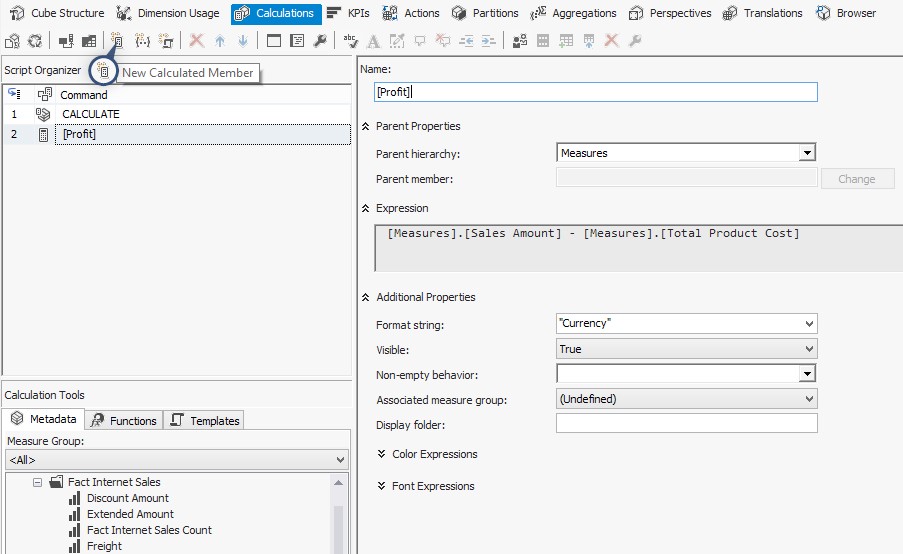
***5***

***.***

Set the format string as

**"Currency"**

.



***6.*** Deploy and process this procedure, and then browse the cube with the **Product Category** hierarchy and a **Profit** measure.

What just happened?

In this extremely simple example, we created a profit column based on the total product cost and the sales amount. The MDX expression used here is very simple; as you can see, you don't need to write the whole **select** statement, you just need to write the expression that calculates or applies conditions.

The MDX expression can be made more complex with the help of functions and metadata information listed in the bottom-left side of the preceding screenshot. You can drag-and -drop functions or measures or dimension hierarchies in the expression box and create the expression easily. There are also some properties for calculated columns, such as font, color, and format string, which can be set in this window.

You can also create Named Sets in this window, which can be a subset of the dimension attribute members, for example, a list of the top 50 customers based on the sales amount. You can write MDX scripts to create a Named Set again.

As you've seen in this section, MDX plays an important role in creating calculations and writing complex query scripts on the SSAS Multidimensional cube, so our recommendation for SSAS developers is to improve their MDX skills. The more professionally you can write MDX, the better you will be able to create more powerful cubes for business requirements.

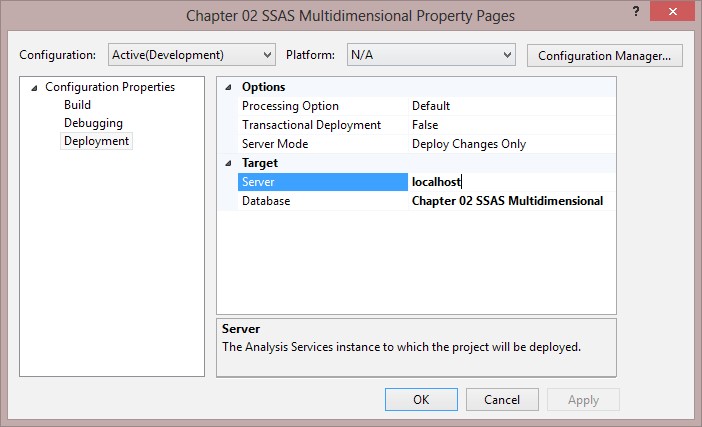
# Deploying and processing

The Analysis Service project contains metadata and data. The metadata of the project can be stored to a server with deployment. Loading data into the SSAS project is called processing. There are multiple ways to deploy a project to the SSAS server, and the processing options also differs based on the method for loading data into the cube.

## Time for action – deploying an SSAS project

You've seen one of the ways of deployment in this chapter, which was deployment from SSDT. Deployment from SSDT can be easily done with a deployment option that can be set in the project's properties. There are other ways for deployment: deployment with **Deployment Wizard** and deployment with XMLA. In this example, we will change some deployment options in the SSDT project and also use the deployment wizard to deploy the project to an SSAS server by performing the following steps:

1. In the **Solution** **Explorer** pane, right-click on a project and select **Properties**.
2. In the project's **Configuration Properties** window, go to the **Deployment** tab. You can see some of the deployment options there, such as the server and database name.



1. Close the project's properties window.
2. Right-click on the project in **Solution** **Explorer** and build the project.
3. Then, go to the **project** folder in Windows Explorer; under the **bin** folder, you will find four files. These files are deployment files that are created after a successful build action.
4. Go to **Start** and under Microsoft SQL Server 2012, and under **Analysis Services**, click on **Deployment Wizard**.
5. In the **Deployment** **Wizard** step, choose the **Chapter 02 SSAS multidimensional.asdatabase** file from the **bin** directory of the project.
6. Set the server name and the database name. Note that the database will be created if the database name doesn't exist in that server. You can set the server as a localhost and the database name as **Chapter 02 SSAS Multidimensional**, which are deployed from the wizard (the database can hold one or more cubes, and each database can also have its own security model and a set of conformed dimensions). ***9.*** You can choose whether you want to keep partitioning on the destination database or you want to overwrite them with partitioning defined in this deployment kit. Leave this option as default.
7. You can also choose to keep the existing roles and membership in the destination database or overwrite them with security settings in the deployment kit. Leave this option as the default.
8. In the **Specify** **Configuration** **Option** tab, you can set the configurations, for example, the impersonation credential to connect to the source database.
9. You can save the deployment script as an XMLA file. Don't create a script for this example.
10. You can also choose a processing method after the deployment; for this example, choose **default processing**.
11. After the deployment and processing step, open SSMS and you will see the new SSAS project created; you can browse the sales cube there to check it.

What just happened?

SSDT has configurations for deployment, such as the deployment server, the database name, and deploy in transaction (step 2). SSDT is a tool for developers and it is good for developers to deploy their projects to the development environment servers. However, in many scenarios, the production environment is not accessible for developers. So, you will have to use other methods of deployment.

**Deployment** **Wizard** is a deployment tool that will be installed with the SSAS service. There are four files required to deploy an SSAS project. These files can be created while building the project (step 4). Each file consists of specific information about deployment. The DBA or the gatekeeper of the production environment can easily run the Deployment Wizard and give the **\*.asdatabase** file path (step 7); follow the steps in the wizard and deploy the project easily.

One of the main benefits of deploying the project with **Deployment** **Wizard** is that you can choose to retain roles and membership or retain partitioning on the destination database (steps 9 and 10). When you conduct deployment through SSDT, the partitioning and roles will be overwritten. This is an important option because in the production environment, partitioning might be set up in the database, and security and role settings might be different from the security configuration in the SSAS project in SSDT.

XMLA is the language for metadata scripts in SSAS. Running the deployment wizard will produce and run an XMLA command on the SSAS server. If you want to overlook all the wizard steps, you can create an XMLA deployment script (step 12), modify it as you wish to make it run on the production environment, and then just pass this script to the DBA and ask them to run it.

### Downloading the example code

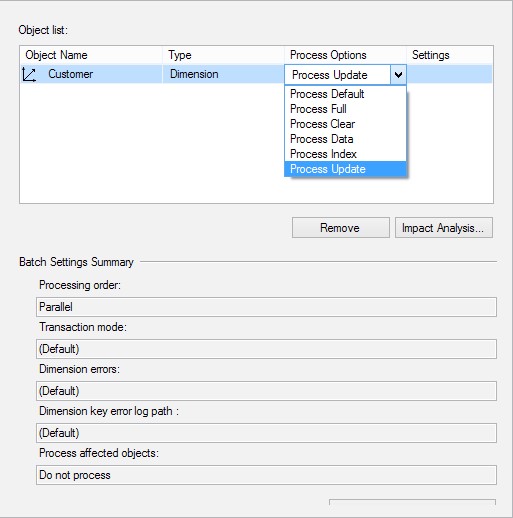
You can download the example code files for all Packt books you have purchased from your account at **http://www.packtpub.com**. If you purchased this book elsewhere, you can visit **http://www.packtpub.com/support** and register to have the files e-mailed directly to you.

## Time for action – processing the data

Deployment will load metadata from an existing Analysis Services project into the SSAS Server, but it won't load data. To load data, another step is required, which is called processing. As there are different requirements to load data, there are multiple types of processing. Sometimes, you just need to populate the changes, but in some cases, such as populating dimensions, you want to populate everything again. In this example, we will go through the processing options by performing the following steps: ***1.*** Open SQL Server Management Studio and connect to the Analysis Services instance.

1. Expand the **Chapter 02 SSAS Multidimensional** database, and then expand the dimensions.
2. Right-click on the **Product** dimension and click on **Process**.
3. In the **Process** window, you can change the process options to **Clear**. Click on the **Clear** option to process and then view the product dimension in the browser; you will see an error message that states that the dimension doesn't contain any hierarchy .

The following screenshot shows the Process Options menu:



1. Process the **Product** dimension again but with the **Process** **Full** option. This time, you will see the product category hierarchy with all members.
2. Right-click on the cube and then click on **Process**. You will see the processing options there as well. This shows that processing can be done in the cube or dimensions level.

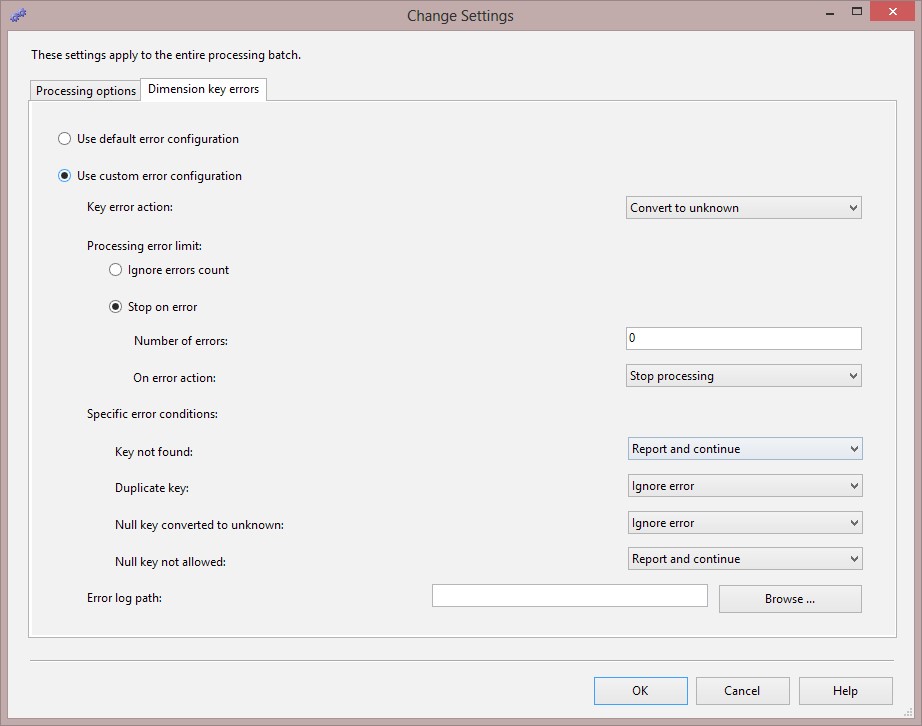
What just happened?

As you've seen in this example, there are different kinds of processing; some of them are the same as dimensions and partitions. Some of them are only for one of the processing types. The following table (available at **http://technet.microsoft.com/en-us/library/ ms174774.aspx**) provides an explanation of the most useful kinds of processing and the SSAS database object that they can be applied to:

|  |  |  |
| --- | --- | --- |
| **Mode** | **Applies to** | **Description** |
| Process Default | Cubes, databases, dimensions, measure groups, mining models, mining structures, and partitions | Detects the process state of database objects and performs the necessary processing to deliver unprocessed or partially processed objects to a fully processed state. If you change a data binding, Process Default will do a Process Full operation on the affected object. |
| Process Full | Cubes, databases, dimensions, measure groups, mining models, mining structures, and partitions | Processes an Analysis Services object and all the objects that it contains. When Process Full is executed against an object that has already been processed, Analysis Services drops all data in the object, and then processes the object. This kind of processing is required when a structural change has been made to an object, for example, when an attribute hierarchy is added, deleted, or renamed. |
| Process Clear | Cubes, databases, dimensions, measure groups, mining models, mining structures, and partitions | Drops the data in the object specified and in any lowerlevel constituent objects. After the data is dropped, it is not reloaded. |
| Process Data | Dimensions, cubes, measure groups, and partitions | Processes data only without building aggregations or indexes. If there is data in the partitions, it will be dropped before repopulating the partition with the source data. |
| **Mode** | **Applies to** | **Description** |
| Process  Add | Dimensions, measure groups, and partitions | For dimensions, this mode adds new members and updates the dimension attribute captions and descriptions.  For measure groups and partitions, this mode adds newly available fact data and a process only to the relevant partitions.  Process Add is not available for dimension processing in Management Studio, but you can write XMLA script performs this action |
| Process  Update | Dimensions | Forces a re-read of data and an update of dimension attributes. Flexible aggregations and indexes on related partitions will be dropped. |
| Process Index | Cubes, dimensions, measure groups, and partitions | Creates or rebuilds indexes and aggregations for all processed partitions. For unprocessed objects, this option generates an error. Processing with this option is required if you turn off Lazy Processing. |
| Process Structure | Cubes and mining structures | If the cube is unprocessed, Analysis Services will process, all the cube's dimensions if necessary. After that, Analysis Services will create only cube definitions. If this option is applied to a mining structure, it populates the mining structure with the source data. The difference between this option and the Process Full option is that this option does not iterate the processing down to the mining models themselves. |
| Process  Clear  Structure | Mining structures | Removes all training data from a mining structure. |

The time taken for processing depends on the structure of data and constraints in the destination database and also depends on the quality of data in the source database; there might be occurrence of some errors. You can catch errors during the processing and do some error configuration. This configuration can be found in the Dimension Designer, cube designer, and also in the **Process** window, as shown in the next screenshot (by clicking on **Change** **Settings**). The next screenshot shows a sample of the **Dimension key errors** settings.

One of the reasons to use error configuration is to deal with errors that occur during processing. For example, you might want to catch errors such as **key not found** and **log** **them**, but you still to perform the processing. So, you can use error configuration to perform settings to cover that requirement. Take a look at the following screenshot, which shows the change settings:



# Summary

In this chapter, you learned some of the benefits of having OLAP cube over the data warehouse. Then we learned how to create cubes and dimensions in the SSAS Multidimensional. We've walked through how to create hierarchies and modify dimensions and measure properties. Then we saw how to create calculated columns in the data source view. Some of the changes had to be made in the cube as a calculation; we used MDX, which is a query language for SSAS Multidimensional, to create calculated measures and a named set. Finally, we saw how to deploy an analysis services project to the SSAS Server and how to load data into the database while processing.

There are still many aspects in Analysis Services multidimensional that need to be studied in a more detailed way. We strongly recommend that you use one of the good books written on SQL Server Analysis Services in order to dive deep in to all the aspects of Analysis Services and learn how to face real-world scenarios with SSAS Multidimensional.