Chapter 4

Data Sources and Data Source Views

What's in this chapter?

- Understanding data sources
- Working with data providers
- Creating Data Source Views (DSVs) with the DSV Wizard
- Enhancing DSVs with the DSV Designer
- Working with DSVs in depth
- Using multiple data sources in a DSV

You have completed Chapter 2 of the book, where you worked hands on with SQL Server Analysis Services (SSAS) 2012 multidimensional tools, and Chapter 3, where you learned the basics of the MDX language and used it to retrieve data from Analysis Services. The next three chapters of the book guide you in the use of the product to design dimensions and cubes.

The traditional approach of dimension and cube design is based on an existing single-source data set. In the real world, you will likely work with multiple relational data sources when you develop business intelligence applications. In this chapter you learn what data sources are in the Analysis Services multidimensional world and how they feed into the creation of Data Source Views (DSVs). These DSVs provide you with a consolidated, single-source view on just the data of interest across one or more data sources you define. The data sources and DSVs form the foundation for subsequent construction of both dimensions and cubes. More than one data source per project is supported as are multiple DSVs. You learn how this infrastructure plays out in this chapter.

Data Sources

For Analysis Services to retrieve data from a source, you need to provide it with information about that source such as its name, the method used to retrieve the data, security permissions needed to retrieve the data, and so on. In Analysis Services, all this information is encapsulated into an object called a *data source*. An Analysis Services database contains a collection of data sources, which holds all the data sources used to build the dimensions and cubes within that database. Analysis Services can retrieve source data from data sources via the native OLE DB provider interface or the .NET Managed Provider interface.

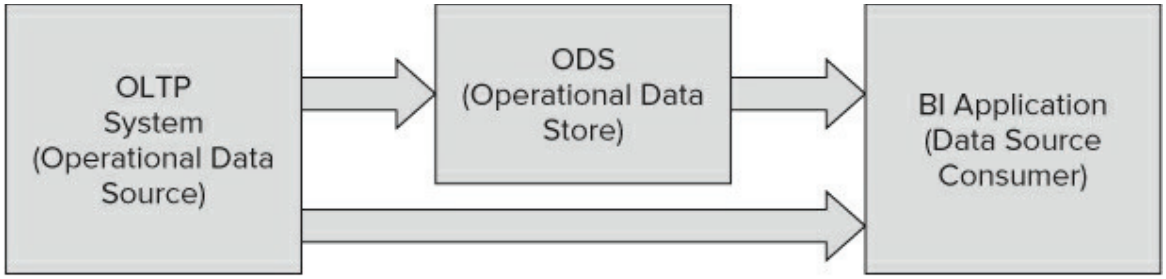
In the simplest case you can have one data source that contains one fact table with some number of dimensions linked to it by joins. That data source is populated by data from an OLTP database and is called an *Operational Data Store* (ODS). Figure 4.1 shows a graphical representation of this data source usage. The ODS is a single entity storing data from various sources so that it can serve as a single source of data for your data warehouse.

Figure 4.1

Data Source path commonly used prior to introduction of the multidimensional model; data is first transformed to a more usable format and stored in the OD.

A variant of data source usage, which is enabled by the Business Intelligence Semantic Model (BISM) multidimensional model, is the ability to take data directly from the OLTP system as input to a business intelligence (BI) application, which is shown in Figure 4.2.

Figure 4.2



Data Sources enabled with BISM multidimensional; data is transformed to a more usable format and stored in the ODS and data can be easily obtained from different data sources without going through the ODS.

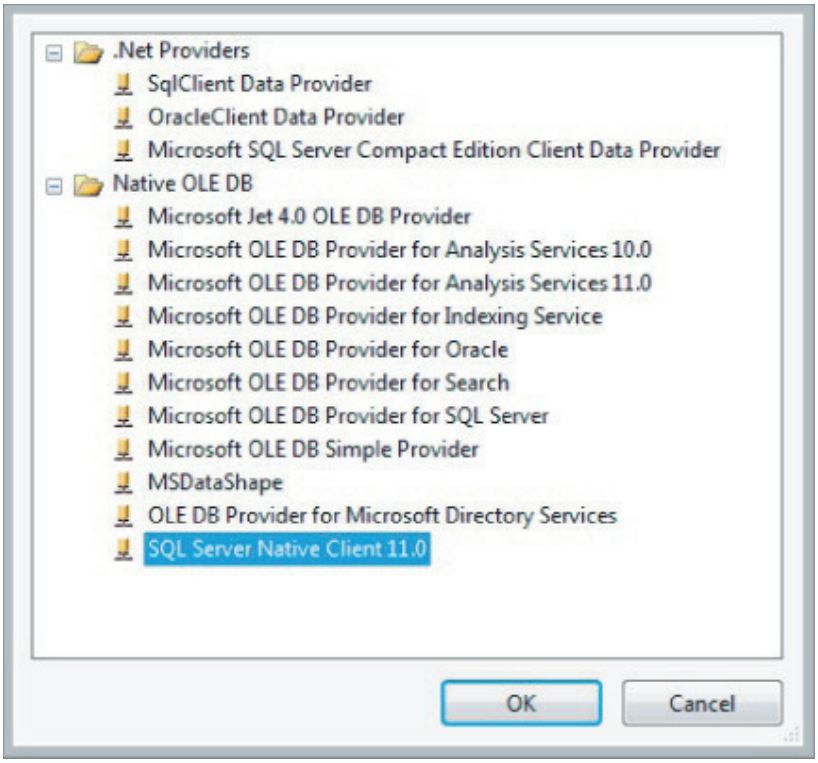
Analysis Services supports building cubes based on multiple fact tables which can come from different data sources. As shown in Figure 4.2, those data sources can come from the operational data store as well as from other places. You can also do things like re-use one cube's measure groups in a different cube. Such measure groups are called *shared objects* and are explained more in Chapter 9. As you can see, the BISM multidimensional model gives you a lot of flexibility when you design your application's data model. This flexibility can be a double-edged sword, however, and we recommend that you understand all the implications of the decision to build models with multiple data sources and very complex schemas. Those implications are discussed in Parts II and III of this book.

Data Sources Supported by Analysis Services

In general, Analysis Services 2012 supports all data sources that expose a connectivity interface through OLE DB or a .NET Managed Provider. This is because Analysis Services uses those interfaces to retrieve the schema information (tables, relationships, columns within tables, and data types) and data it needs to work with those data sources. If the data source is a relational database, then by default it uses standard SQL to query the database. Analysis Services uses a "cartridge" mechanism that enables it to translate its standard SQL language to the SQL dialects of different relational database systems. Cartridges are, in essence, XSLT transforms that are applied to the standard SQL stored internally to Analysis Services in an XML format.

Officially, Analysis Services 2012 supports specific relational data sources. The major relational data sources supported by Analysis Services include Microsoft SQL Server and Parallel Data Warehouse, IBM's DB2, Teradata, Oracle, Sybase, and Informix. Figure 4.3 shows various data sources supported by Analysis Services 2012 on a machine that has SQL Server 2012 installed. For a specific data source, you may need to install additional client components of the data provider so that the OLE DB provider and/or .NET provider for that specific data source is available on your machine. These client components should not only be available on the development machine where you use SQL Server Data Tools (SSDT) to design your database, but they also must be available on the server machine where the Analysis Services instance runs. For relational databases DB2 and Oracle, it is recommended you use Microsoft's OLE DB data provider for Oracle or DB2 (Microsoft's DB2 driver is available in the SQL Server Feature Pack) instead of the OLE DB providers provided with those databases. Install appropriate connectivity components from Oracle and IBM's DB2 on your machine in addition to the OLE DB providers from Microsoft.

Figure 4.3



In Chapter 2 you used the Data Source Wizard to create a data source that included impersonation information. You use that Analysis Services project created in Chapter 2 for the illustrations and examples in this chapter.

In addition to providing impersonation information, you can optionally specify additional connection properties; such as query time out for connection, isolation level, and maximum number of connections; in the Connection Manager dialog, as shown in Figure 4.4, at the time you create your data source. Alternatively, you can define additional connection properties after the data source has been created using the Data Source Designer dialog, as shown in Figure 4.5. The isolation-level property has two modes: Read Committed and Snapshot. By default, Read Committed is used for all the data sources. The Snapshot isolation mode, which is supported by SQL Server and Oracle relational data sources, ensures that the data read by Analysis Services is consistent across multiple queries sent over a single connection. This means that if the data in the relational data source changes while multiple queries are sent by Analysis Services to that relational data source, all the queries see the same data that was seen by the first query. Any changes to the data between the time the first query and Nth query were sent over the same connection are not included in the results of subsequent queries. The specified connection properties, including isolation level, are stored and applied whenever a connection is established to a specific data source.

Figure 4.4

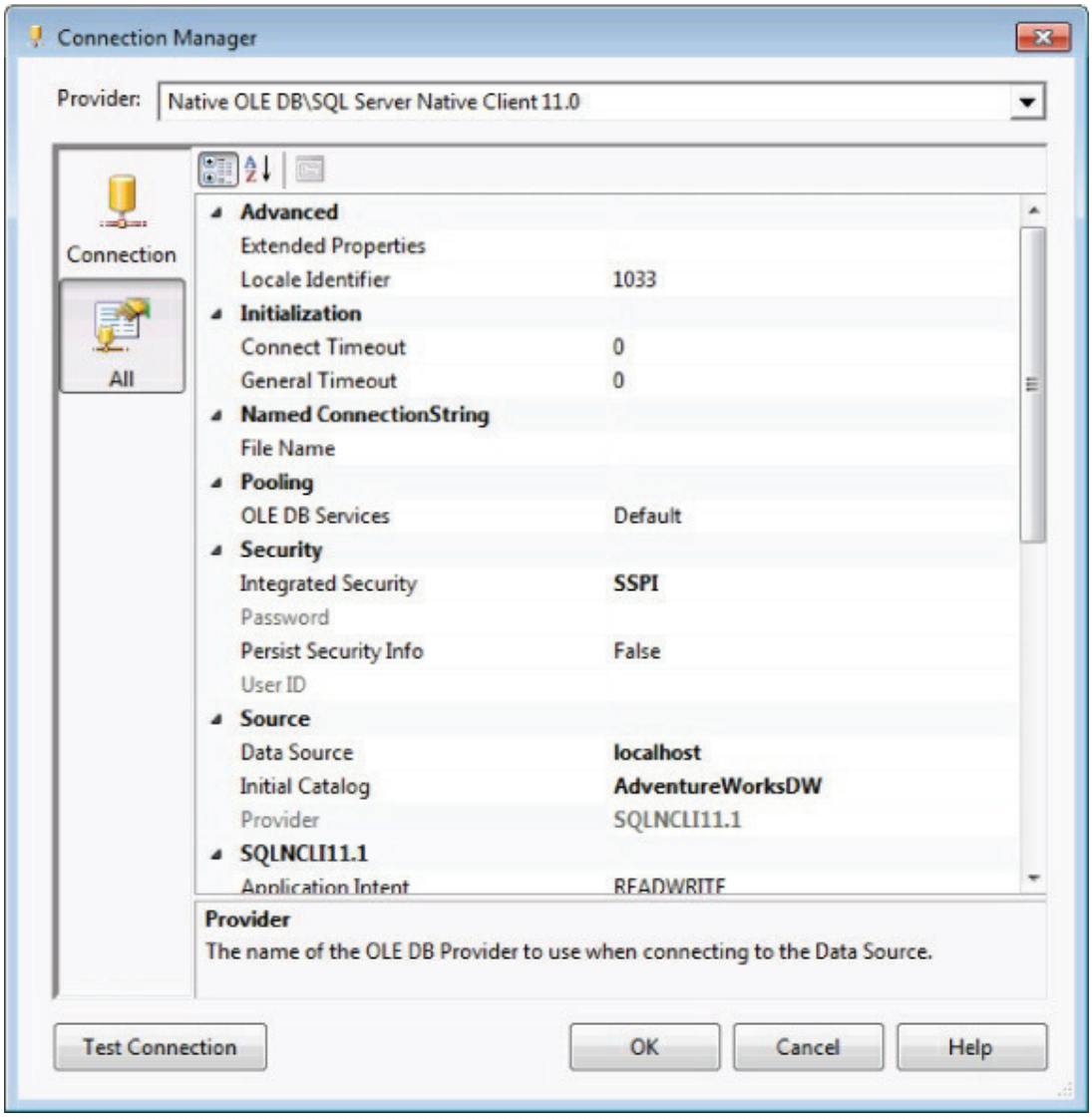
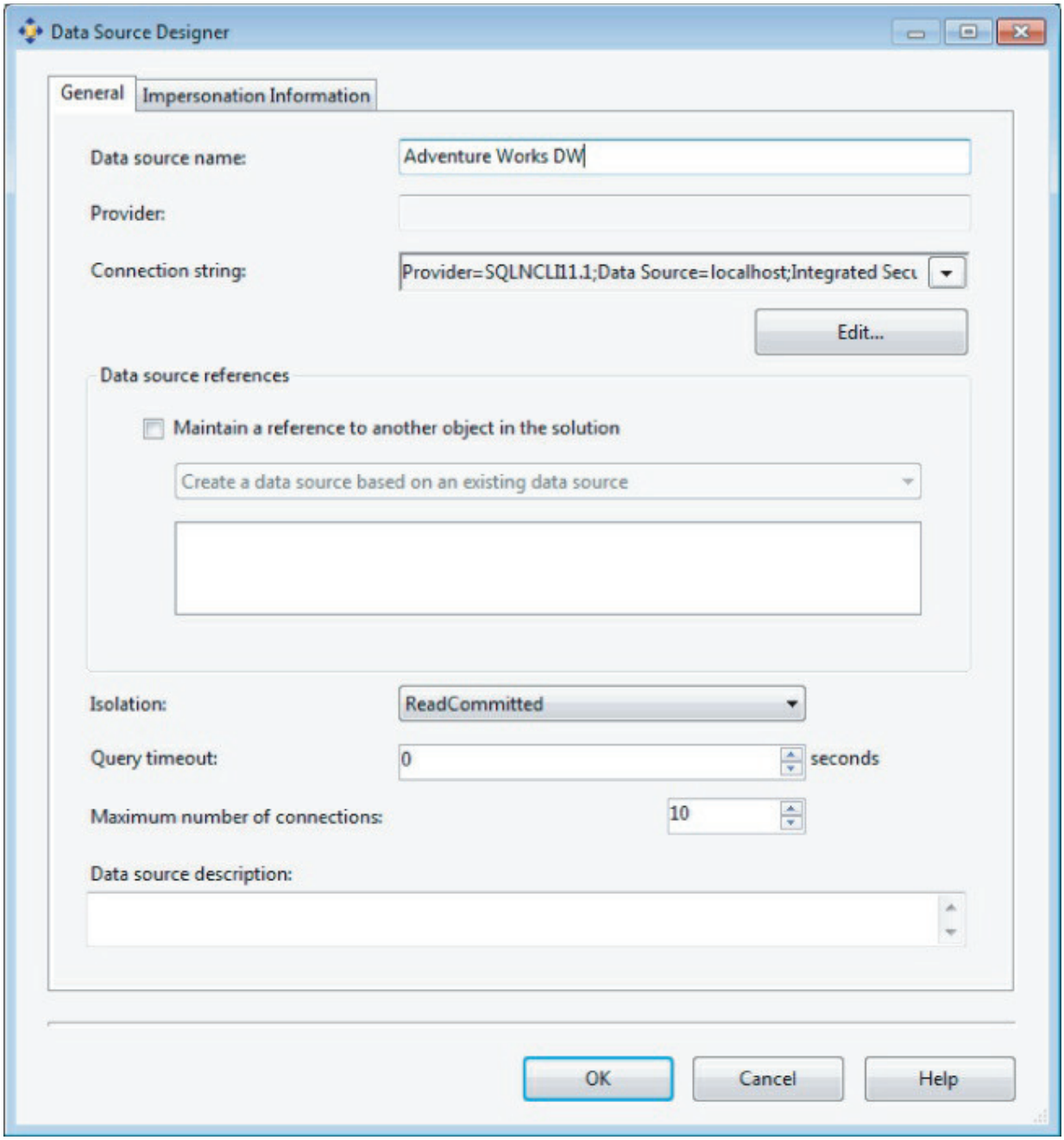

Connection Manager
Provider:
Native OLE DB\SQL Server Native Client 11.0
Connection
All
Advanced
Extended Properties
Locale Identifier
1033
Initialization
Connect Timeout
0
General Timeout
0
Named ConnectionString
File Name
Pooling
OLE DB Services
Default
Security
Integrated Security
SSPI
Password
Persist Security Info
False
User ID
Source
Data Source
localhost
Initial Catalog
AdventureWorksDW
Provider
SQLNCLI11.1
SQLNCLI11.1
Application Intent
READWRITE
Provider
The name of the OLE DB Provider to use when connecting to the Data Source.
Test Connection
OK
Cancel
Help

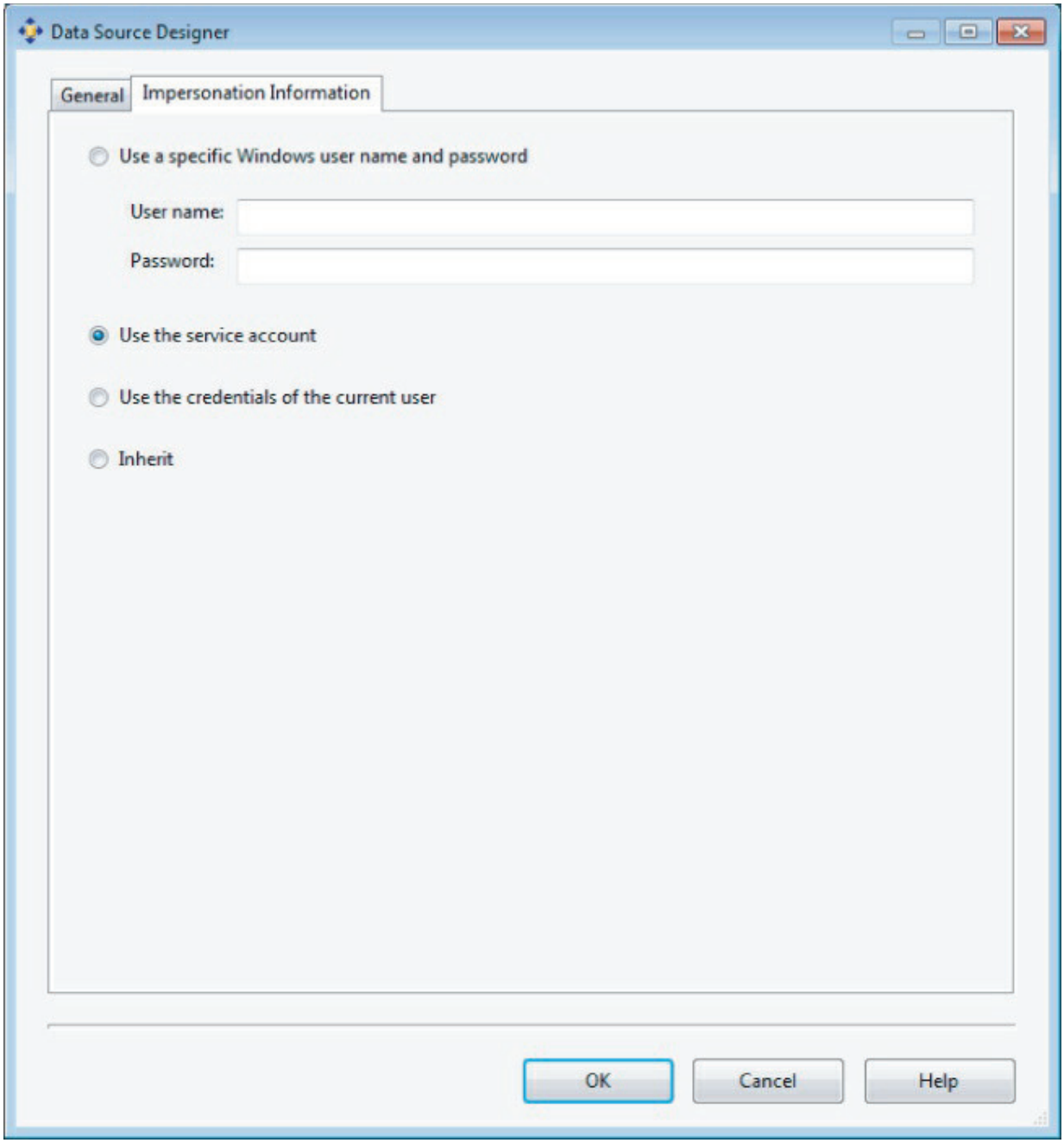
Figure 4.5



The Data Source Wizard enables you to create data sources based on an existing data source connection. This means that multiple Analysis Services databases can share a single data source connection. The wizard also enables you to establish connections to objects within the current Analysis Services project, such as establishing an OLE DB connection to a cube created in the project. Such a connection is typically useful when creating mining models (discussed in Chapter 12) from cubes.

The Impersonation Information tab in the Data Source Designer has four options, as shown in Figure 4.6. You briefly learned about these options in Chapter 2. At development time, SSDT uses the current user's credentials to connect to the relational data source and retrieve data. However, after the Analysis Services project deploys it needs credentials to connect to and retrieve data. Here you specify the impersonation information that Analysis Services uses to make that connection. The following list gives more details on the four options and when to use them:

Figure 4.6



- **Use a specific Windows username and password:** You typically would choose this option when the account used by the Analysis Services instance does not have permissions to access the relational backend. When you select this option, you need to specify a Windows username and password that Analysis Services can use to connect to the relational backend. For security reasons the password and connection string used to connect to the relational backend are encrypted before they are sent to the Analysis Services instance.
- **Use the service account:** This option uses the credentials of the Analysis Services service startup account to connect to the relational backend. You need to make sure that the account has permission to access the relational data source.
- **Use the credentials of the current user:** Typically, you select this option for data mining. Use this option for out-of-line bindings, DMX OPENQUERY statements, local cubes, and mining models. Do not select this option when you connect to a relational backend for processing, ROLAP queries, remote partitions, linked objects, and synchronization from target to source.
- **Inherit:** This option instructs Analysis Services to use the impersonation information specified in the parent Analysis Services database object.

.NET Versus OLE DB Data Providers

Most data sources support one or both of the two types of data providers that Analysis Services can use to connect to external data: OLE DB providers or .NET Framework data providers. The OLE DB standard defines a set of COM interfaces for data access. The

.NET Framework data provider interface is similar to OLE DB although providers using that interface are implemented in .NET managed code rather than native COM. Analysis Services has the capability to use OLE DB or .NET Framework data providers to access data sources ranging from flat files to large-scale databases such as SQL Server, Oracle, Teradata, DB2, Sybase, and Informix. Analysis Services retrieves data from the data sources using the chosen provider's interface (OLE DB or .NET) when processing Analysis Services objects. If any of the Analysis Services objects are defined as ROLAP, the provider is also used to retrieve data at query time. Analysis Services also provides the ability to update the data in a multidimensional model. This feature is called *writeback*. Analysis Services also uses the provider interfaces to update the source data during writeback. (You learn about writeback in Chapter 9.)

.NET Framework Data Providers

The Microsoft .NET Framework and programming languages that use the framework run in the Common Language Runtime (CLR) environment. The relationship between .NET languages and the CLR is analogous to that of the Java language and the Java Runtime (the JVM or Java Virtual Machine). The .NET Framework is a huge class library that provides support for almost every kind of functionality that most application programs need and does so in the context of managed code. The term *managed code* refers to the fact that memory is managed by the CLR and not the application program. You can write your own managed code provider for your data source, or you can leverage existing .NET Framework data providers. With the installation of SQL Server 2012, you have .NET providers that support Microsoft SQL Server and Oracle (refer to Figure 4.3). If you need to use a different relational data source and it has a .NET provider, you can install it and use that provider. In the Connection Manager page of the Data Source Wizard, you can choose from the installed .NET providers to connect to your data source.

OLE DB Data Providers

OLE DB is an industry-standard application programming interface that defines a set of Component Object Model (COM) interfaces for accessing data from various data sources. The OLE DB standard was created for client applications to have a uniform interface from which to access data. Such data can come from a wide variety of data sources such as Microsoft Access, Microsoft Project, and various database management systems.

Microsoft provides a set of OLE DB data providers to access data from several widely used data sources. This set of providers includes SQL Server OLE DB provider, Oracle OLE DB provider, and the OLE DB provider for ODBC. These OLE DB providers have been included in the operating system since Windows Server 2003 and Windows XP SP2 as a component known as Windows Data Access Components (WDAC). (This component was formerly known as Microsoft Data Access Components [MDAC] and was renamed with the release of Windows Vista.) Even though the interfaces exposed by the providers are common, each provider is different in the sense they have specific optimizations relevant to the specific backend data source. OLE DB providers, being implementations of the OLE DB COM interfaces, are written in unmanaged code.

The Microsoft OLE DB provider for SQL Server included with WDAC provides basic connectivity to SQL Server for users of the Windows family of operating systems. You don't need SQL Server installed to use that provider. If you do install SQL Server, you have the option to install the SQL Server Native Client (SNAC), which is a single component containing enhanced versions of the SQL Server ODBC provider and the SQL Server OLE DB provider. The SQL Server Native client also provides the SQL Server team the ability to enhance the functionality of its native connectivity components on a different ship schedule than the providers that ship in the operating system as part of WDAC and are tied to OS releases. The SQL Server 2012 release provides version 11 of SQL Server Native Client and is used in the data source connection string (refer to Figure 4.5). Analysis Services provides the ability to connect to any data source that provides an OLE DB interface, including the Analysis Services OLE DB provider.

The Trade-Offs

Versions of Analysis Services prior to Analysis Services 2005 supported connecting to data sources through OLE DB providers only. Analysis Services 2005 and later have much tighter integration with the .NET Framework and support connections via both OLE DB and .NET data providers. If you deployed the .NET Framework across your entire organization, you should use the .NET providers to access data from relational data sources. You might encounter a small amount of performance degradation using the .NET provider; however, the uniformity, maintainability, inherent connection pooling capabilities, and security provided by .NET data providers are worth taking the small hit on performance. If you are concerned about the fastest possible performance, use OLE DB providers for your data access.

Data Source Views

Data Source Views (DSVs) enable you to create a logical view of the tables and views from your data source. They are a powerful tool. You can, for example, filter out any tables not involved in your data warehouse design. In this way, you can exclude from the virtual workspace system tables and other tables not pertinent to your efforts. You can create DSVs that contain tables from multiple data sources, which you learn about later in this chapter in the “Multiple Data Sources Within a DSV” section.

You need to create DSVs in your Analysis Services databases because cubes and dimensions are created from a DSV rather than directly from data source objects. When you create a DSV, the DSV Wizard retrieves the schema information from the data source

including relationships. This schema information is stored in the DSV and helps the Cube and Dimension Wizards identify fact and dimension tables as well as hierarchies. If the right relationships do not exist in the data source, you can create them within the DSV. Taking the time to create an effective DSV ultimately pays for itself in terms of speeding up the design of your data warehouse.

You use the DSV Wizard to create DSVs. After you create a DSV, you can perform operations on it, such as adding or removing tables, specifying virtual primary keys, and establishing relationships using the DSV Designer. You learn more about the DSV Wizard and the DSV Designer and how to work with them in the following sections.

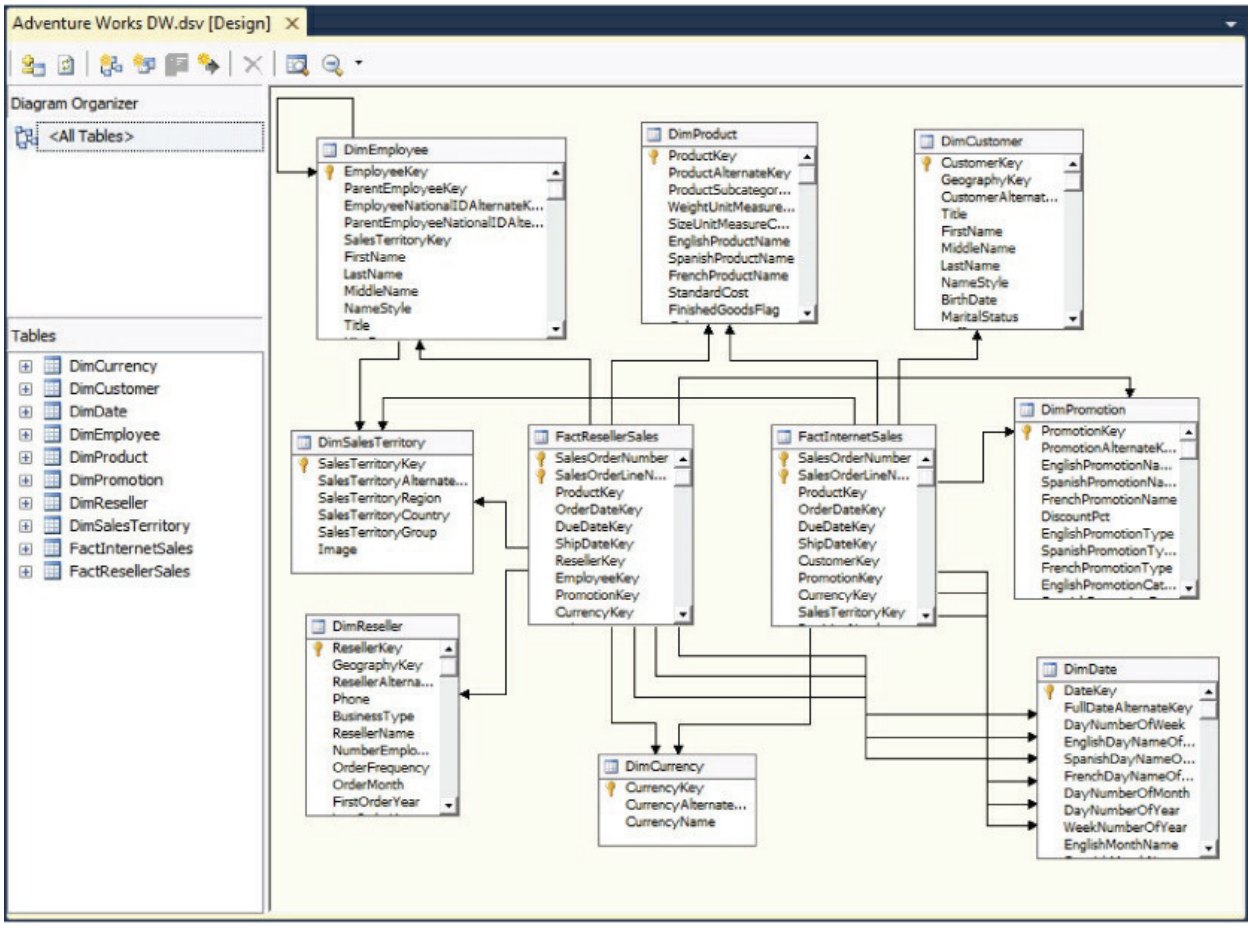
DSV Wizard

In Chapter 2, you used the DSV Wizard to create a DSV based on the sales fact and related dimension tables in the Adventure Works DW relational database. You need a data source object to create a DSV. If you had not created a data source object in your database before running the DSV Wizard, you can do so from within the DSV Wizard's Select a Data Source page by clicking the New Data Source button. In addition, the DSV Wizard enables you to restrict your DSV to specific schemas as well as filter tables included in the DSV, which helps you to work more effectively using only the tables you need in your DSV.

DSV Designer

The DSV Designer contains three panes, as shown in Figure 4.7. The rightmost pane contains a graphical view of all the tables in the DSV. This pane is called the *diagram pane*. Relationships between tables are represented by lines connecting two tables with an arrow at the end. Primary key columns are indicated by a small key icon to the left of the column name.

Figure 4.7

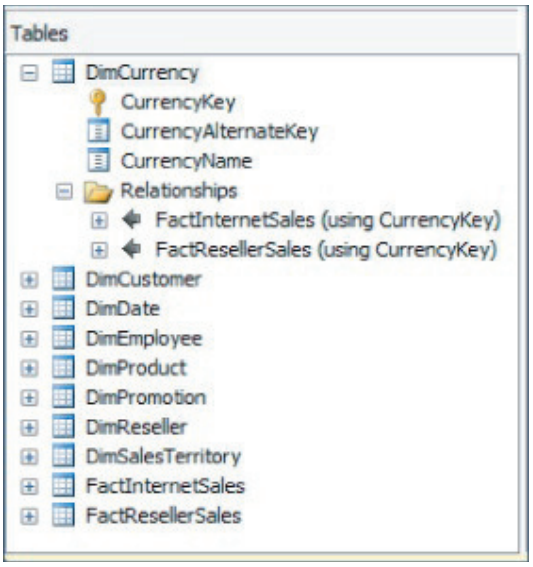


The top-left pane is the *Diagram Organizer*, which is helpful in creating and saving concise views of subsets of the objects in your DSVs. This is especially helpful when DSVs contain many tables. When a DSV contains, for example, more than 20 tables it can be difficult to visualize the complete DSV in the diagram pane. When you have a large number of tables, you likely perform operations only on a subset of them at any given time. The Diagram Organizer is a handy way to create additional diagrams that include just such subsets of relevant tables. Operations done on tables within a diagram are reflected in real-time in the entire DSV. By default one diagram called All Tables contains all the objects in the DSV.

Figure 4.7 shows part of the default All Tables diagram that is created when you complete the DSV Wizard.

The lower-left pane of the DSV Designer is called the *Tables pane* and shows a tree view of all the tables in the DSV along with their relationships to other tables. Figure 4.8 shows the Tables pane with detailed information of the DimCurrency table. You can see the primary key of the DimCurrency table, CurrencyKey, which is distinguished by a key icon. In addition, there is a folder that contains the relationships between the DimCurrency table and other tables in the DSV. If you expand the Relationships folder (as shown in Figure 4.8) you can see that the DimCurrency table joins to the FactInternetSales and FactResellerSales tables through the CurrencyKey column (indicated in parentheses).

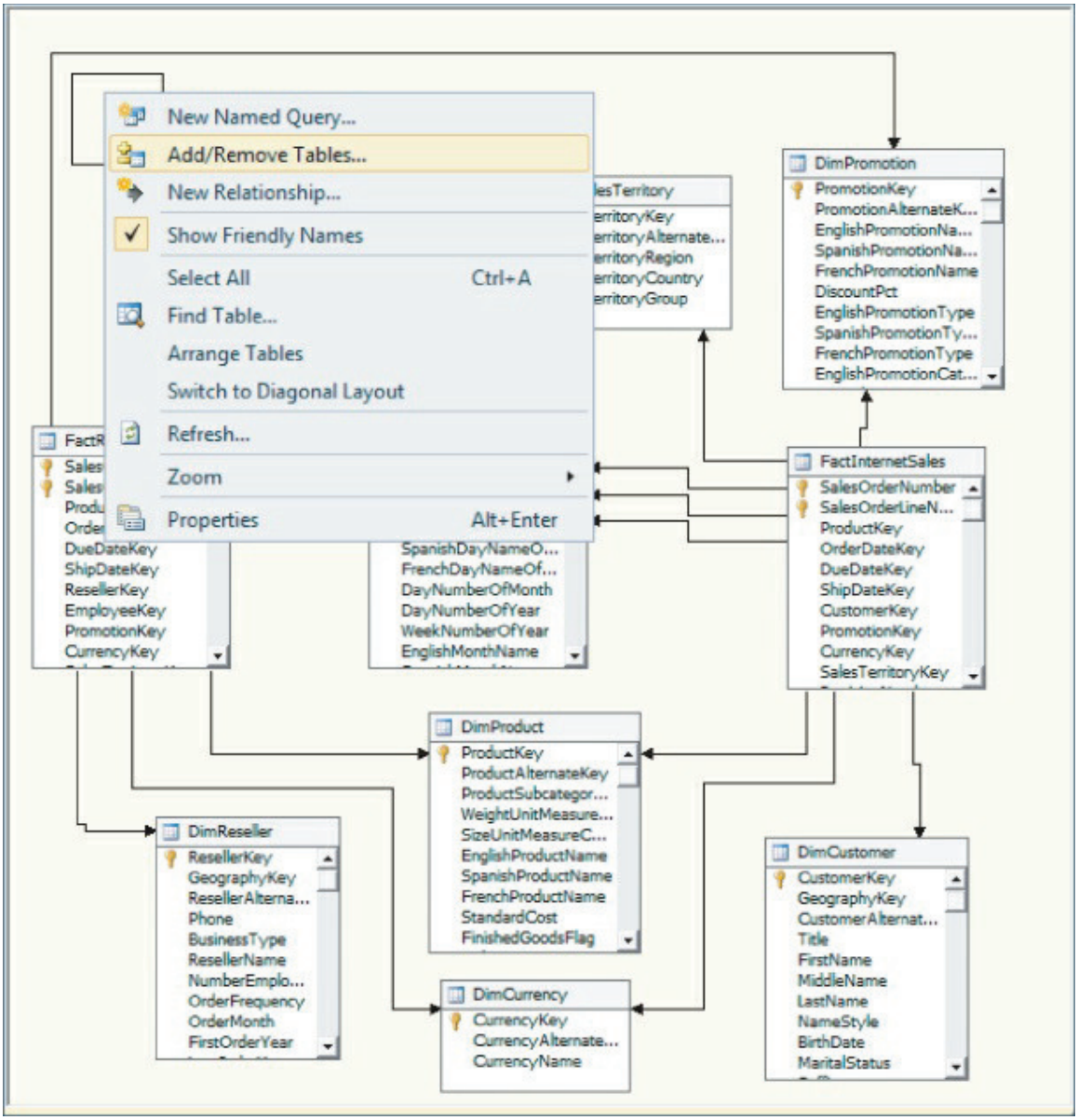
Figure 4.8



Adding/Removing Tables in a DSV

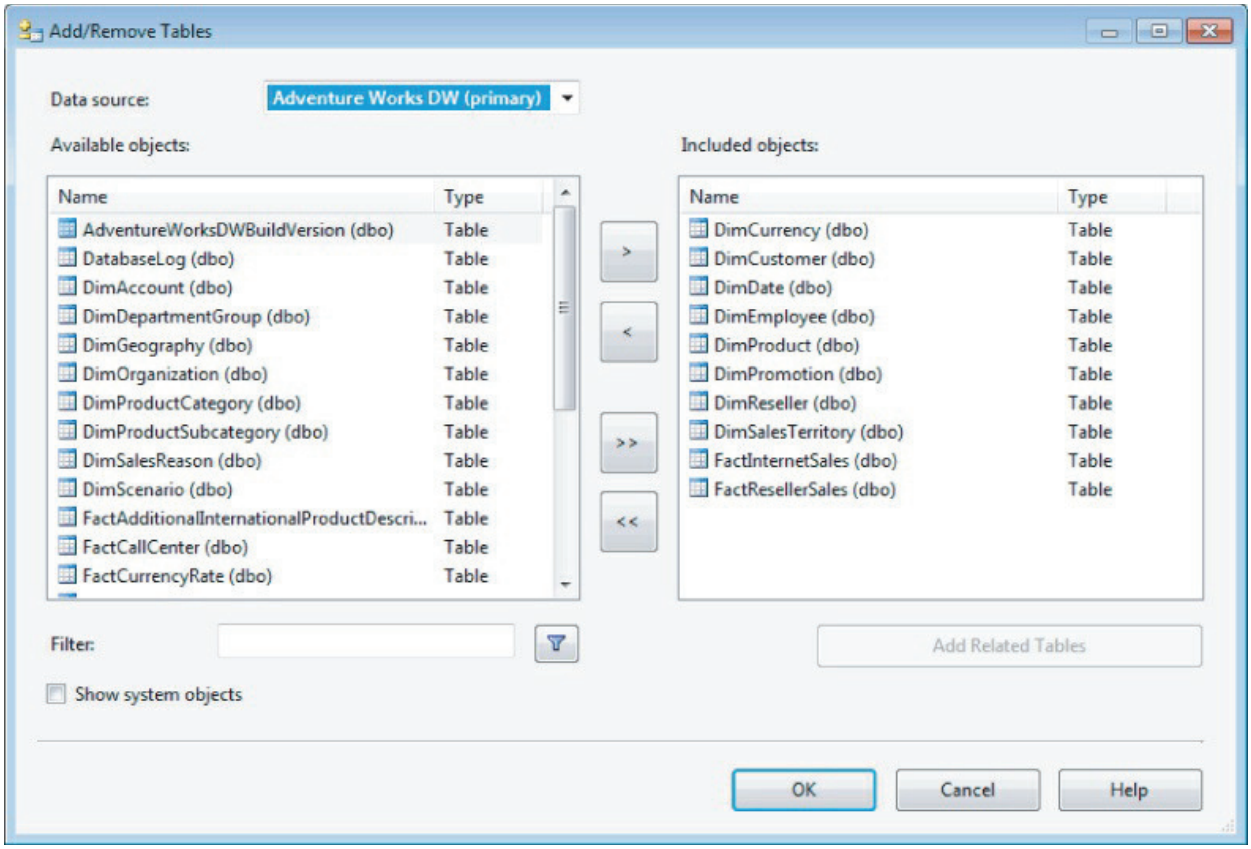
The DSV Wizard creates DSVs. It generates the initial DSV based on what it can discover from the schema of the source data. The real power of DSVs, though, comes from the ability to modify the initial DSV to align with the needs of the BI solution you build. You use the DSV Designer to tap into that power. To modify existing tables in the DSV, right-click the diagram pane, and select Add/Remove Tables, as shown in Figure 4.9.

Figure 4.9



This invokes the Add/Remove Tables dialog shown in Figure 4.10. Using this dialog you can add additional tables to the DSV by moving tables from the Available objects list to the Included objects list or remove existing tables by moving them from the Included objects list to the Available objects list. You can also remove a table in the DSV Designer's diagram pane or Table pane using the following steps:

Figure 4.10



1. Select the table to be deleted.

2. Right-click the table, and on the context menu select Delete Table from DSV.

3. In the confirmation dialog that appears, click OK.

Specifying Primary Keys and Relationships in the DSV

It is likely that you will encounter underlying databases without the primary key to foreign key relationships that you need in place for preparation of data for analysis — that is, for building dimensions and cubes. The DSV Wizard can extract the primary keys and relationships specified in the underlying relational database and add them to your DSV. But perhaps some of the OLTP data you use does not have primary keys and relationships specified for the relevant tables. The DSV Designer provides you with the functionality to specify logical primary keys for the tables that do not have primary keys defined for them. In this way you can effectively modify or add new relationships between the tables in the DSV.

To specify a logical primary key for a table, do the following in the DSV Designer:

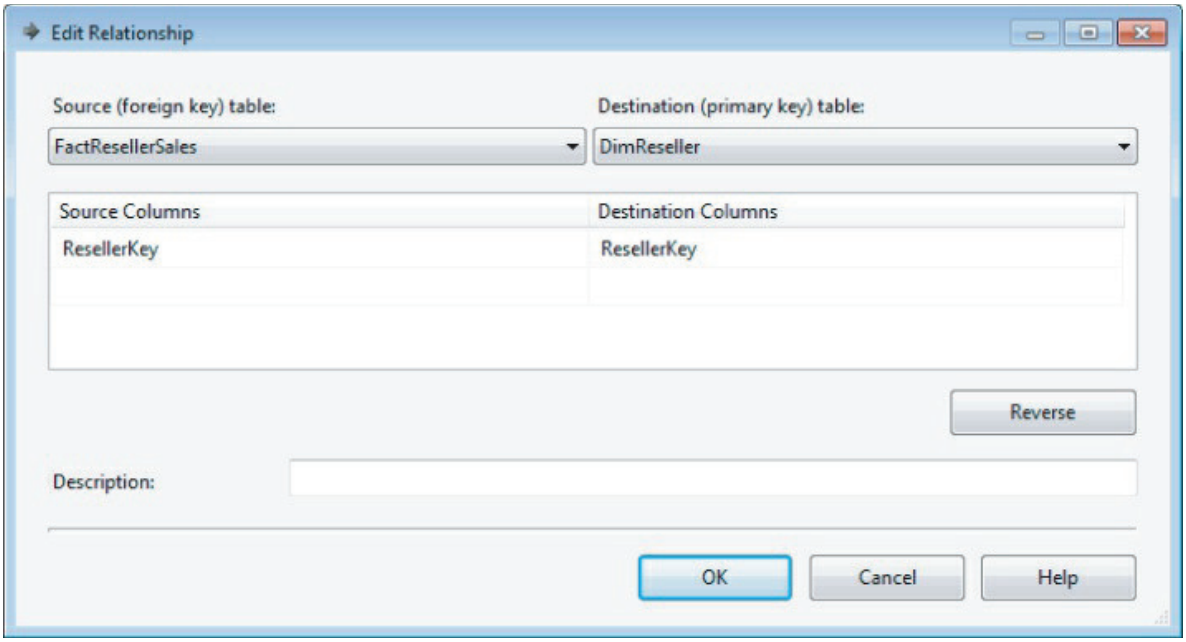
1. Select the column(s) in the table that you want to specify as the logical primary key. If there is more than one column that forms the primary key, you can select multiple columns by holding down the Ctrl key while selecting.

2. Right-click and select Set Logical Primary Key. When there is a relationship between two tables, Table1 and Table2, you typically have columns A in Table1 and B in Table2 involved in the join. If column B is the primary key in Table2, column A in Table1 is referred to as the foreign key. An example would be a Sales fact table that has a Product ID column that joins with the Product ID column in the Products dimension table. To specify relationships between the tables in this hypothetical example, you would use the following steps.

A. Select column A in Table1.

B. With column A selected, drag and drop it to column B in Table2. This forms a relationship between Table1 and Table2. A line will be created between these two tables with an arrow pointing toward Table2. If you double-click this line, you see the details of the relationship — the tables involved in the relationship and the columns used for the join. Figure 4.11 shows the relationship between the FactResellerSales and DimReseller tables. You can modify the relationship using this Edit Relationship dialog by either changing the columns involved in the join or by adding additional columns involved in the join.

Figure 4.11



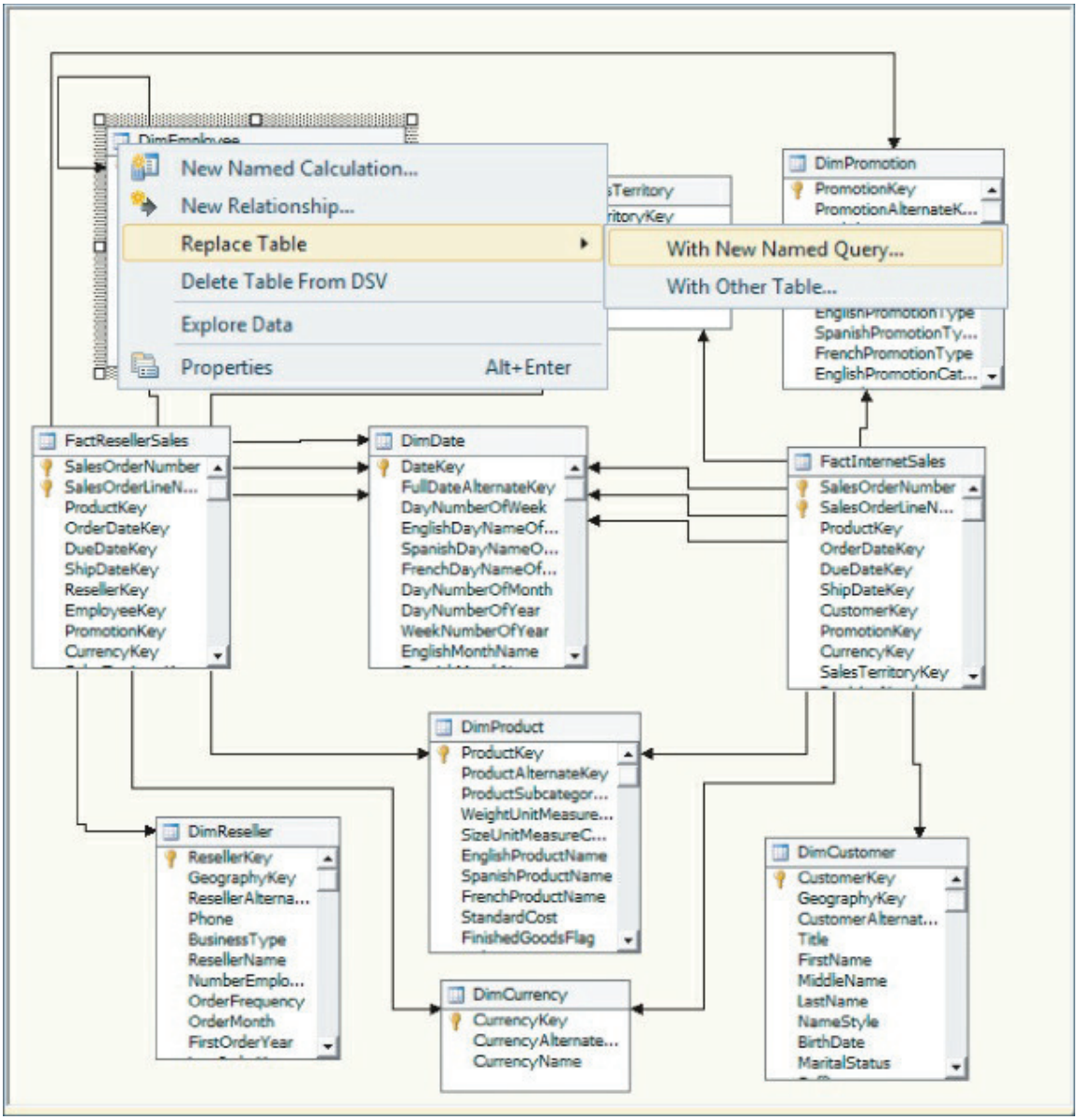
You can also create a new relationship by right-clicking a table and selecting New Relationship. You will be asked to specify the source and destination columns that make up the relationship in the Create Relationship dialog, which is similar to the Edit Relationship dialog shown in Figure 4.11. You can also add a description for the relationship.

All operations that you can accomplish in the diagram pane such as specifying primary keys or using drag and drop to create relationships you can also do using the Tables pane of the DSV Designer.

Customizing Your Tables in the DSV Designer

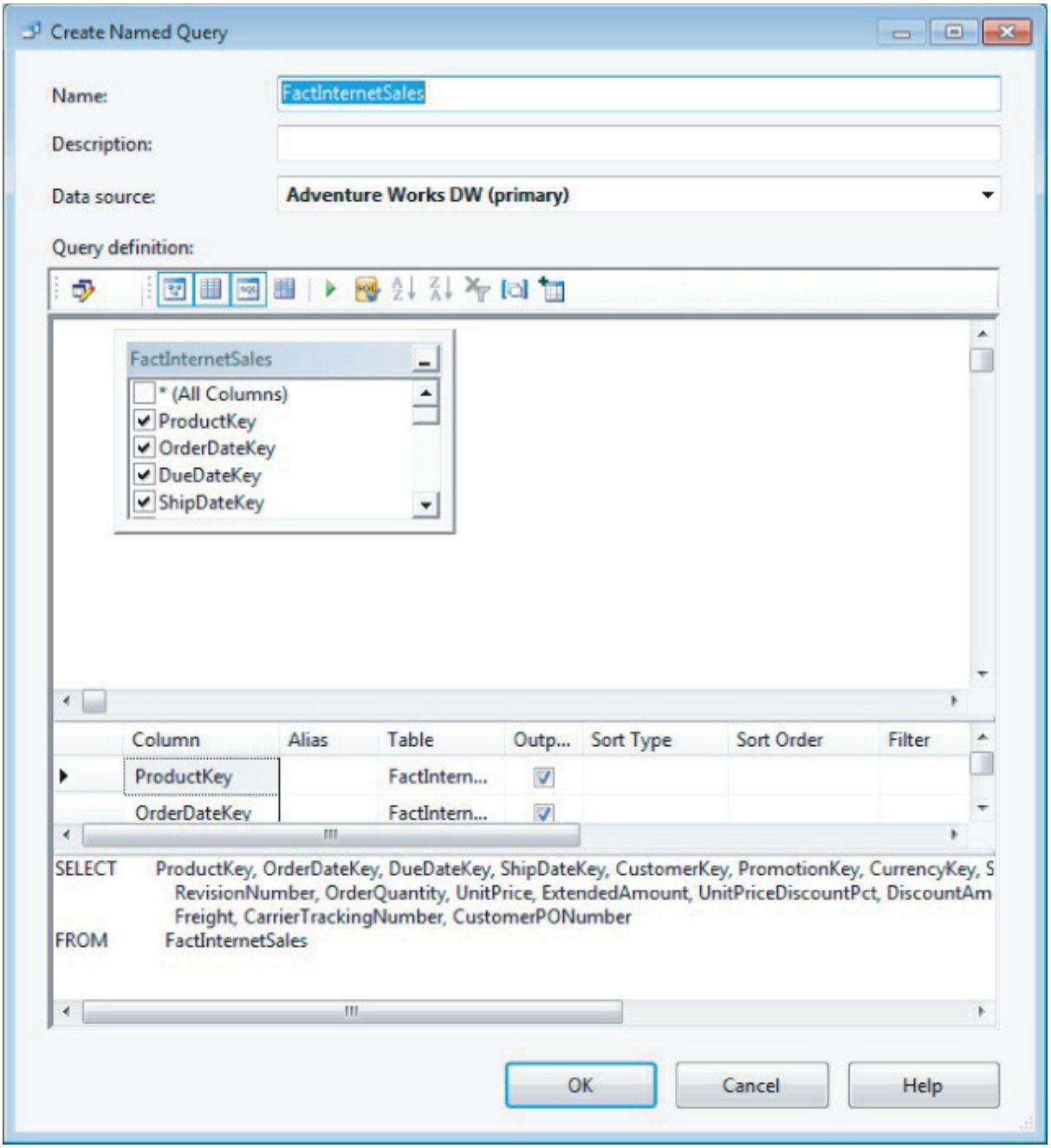
When building your data warehouse, you often want to select a subset of all the columns in a table or restrict the fact table rows based on some specific criteria. Or you might want to merge columns from several tables into a single table. You can do all these operations by creating views in the relational database. In some cases, doing these operations on the relational database side makes sense but in some cases you can't. For example, you may build your model from source data that you have only read-only access to. Or you may build your model using data from different sources. For these cases, Analysis Services provides the functionality to perform these operations within the DSV using a *Named Query*. You can invoke the Named Query editor by right-clicking a table in the diagram pane and selecting Replace Table ⇒ With New Named Query, as shown in Figure 4.12. If you want to add a named query-based table without replacing an existing table in your DSV, right-click in the diagram pane of the DSV Designer but not on a table, and select New Named Query.

Figure 4.12



The Create Named Query dialog is shown in Figure 4.13. In this dialog you can add tables from the data source, select specific columns from a table, and apply restrictions or filters to the rows of a table. A SQL query is created based on your selections and displays in the SQL pane in the editor. If you're a SQL wizard, you can enter or paste a valid SQL query directly into the SQL pane. You can then execute the query to make sure the query is correct. The results from the underlying relational database are then visible in a new pane beneath the SQL pane. Click OK after you form and validate your query. The table is now replaced with results from the Named Query you have specified.

Figure 4.13



In certain instances you might want to create a new column in a table. For example, you may want to create a column that contains the full name of an employee based on the first name, middle name, and last name. One way to accomplish this task would be to replace the entire table with a named query and write the appropriate SQL to include this additional column. However, Analysis Services provides a simpler way to do the same operation. Right-click the Employee table, and select New Named Calculation, as shown in Figure 4.14. This action invokes the Create Named Calculation dialog, as shown in Figure 4.15. To add a column called FullName to the Employee table, you just need to combine the FirstName, MiddleName, and LastNamecolumns. You can type the expression for this named calculation in the Expression pane, as shown in Figure 4.15, and then click the OK button.

Figure 4.14

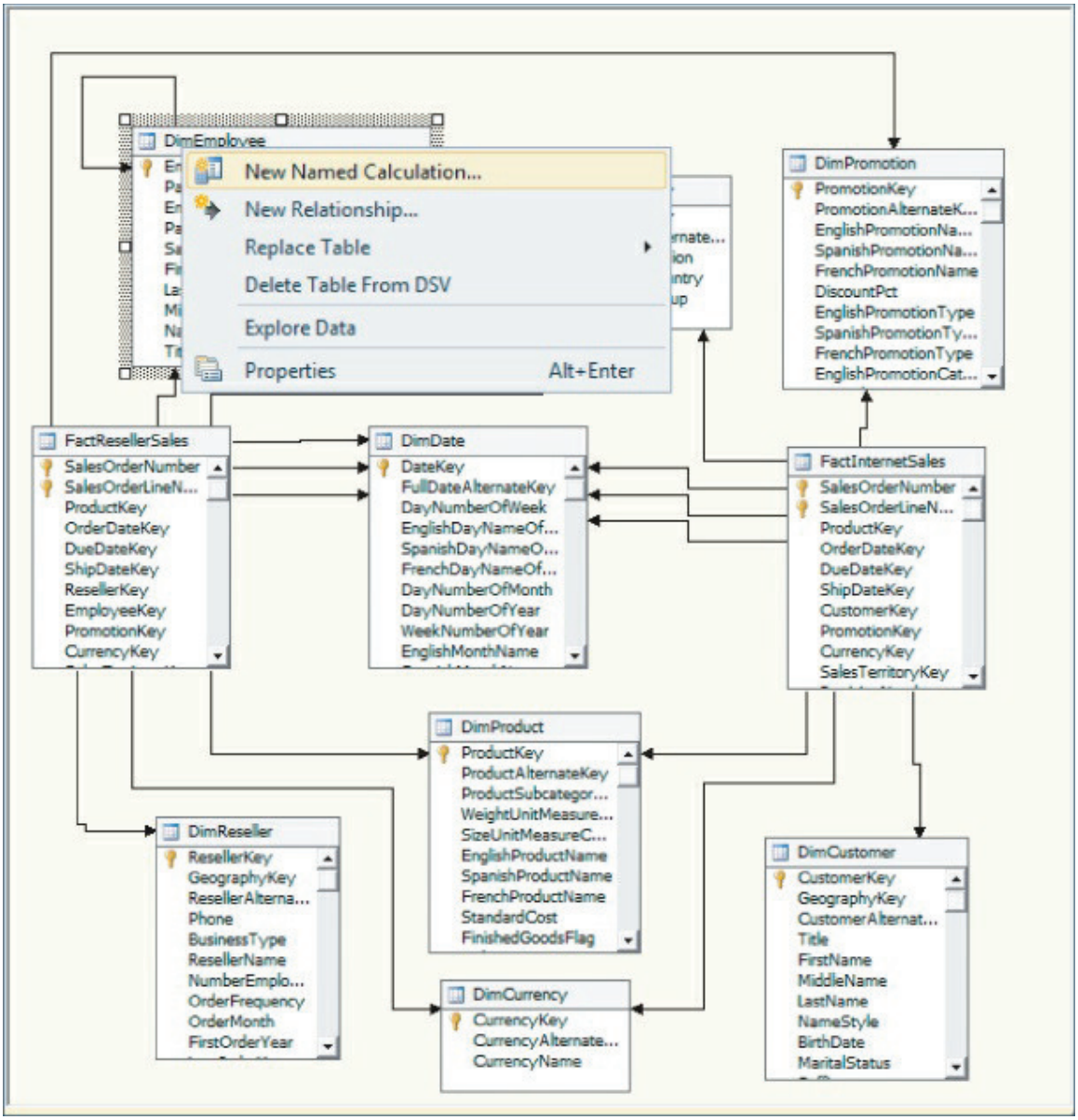
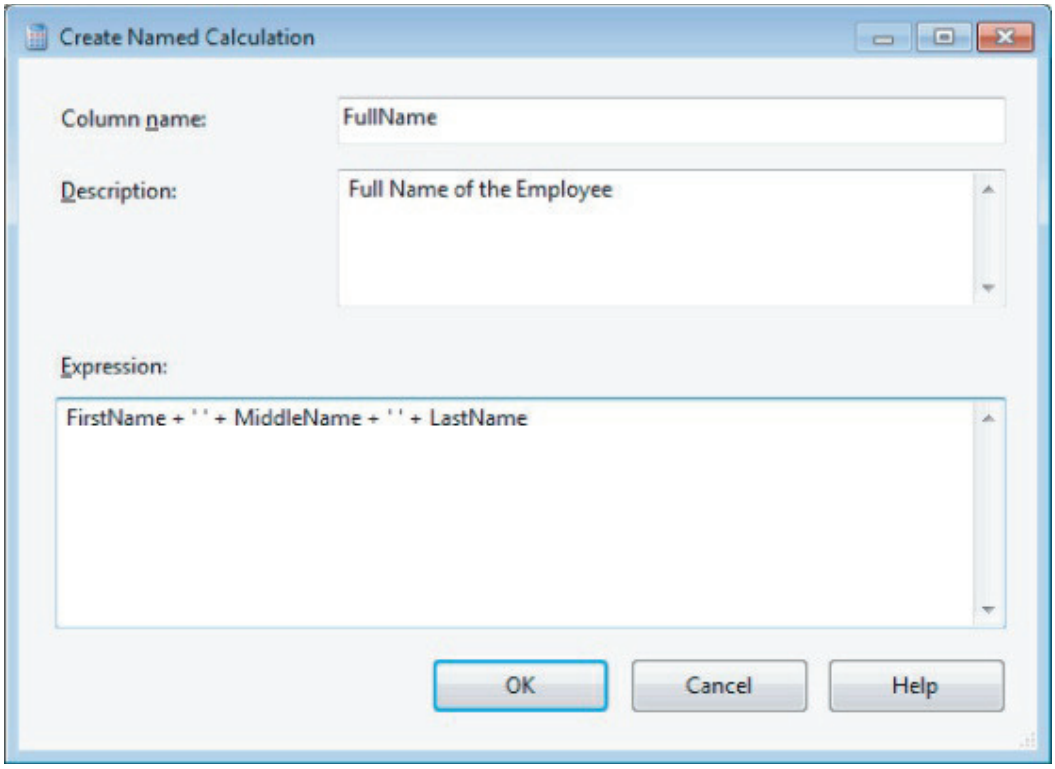
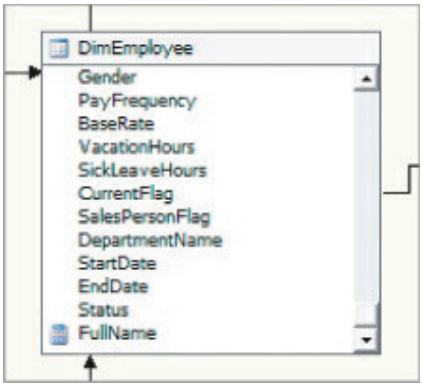

DimEmployee
New Named Calculation...
New Relationship...
Replace Table
Delete Table From DSV
Explore Data
Properties
Alt+Enter
DimPromotion
PromotionKey
PromotionAlternateK...
EnglishPromotionNa...
SpanishPromotionNa...
FrenchPromotionName
DiscountPct
EnglishPromotionType
SpanishPromotionTy...
FrenchPromotionType
EnglishPromotionCat...
FactResellerSales
SalesOrderNumber
SalesOrderLineN...
ProductKey
OrderDateKey
DueDateKey
ShipDateKey
ResellerKey
EmployeeKey
PromotionKey
CurrencyKey
DimDate
DateKey
FullDateAlternateKey
DayNumberOfWeek
EnglishDayNameOf...
SpanishDayNameO...
FrenchDayNameOf...
DayNumberOfMonth
DayNumberOfYear
WeekNumberOfYear
EnglishMonthName
FactInternetSales
SalesOrderNumber
SalesOrderLineN...
ProductKey
OrderDateKey
DueDateKey
ShipDateKey
CustomerKey
PromotionKey
CurrencyKey
SalesTerritoryKey
DimProduct
ProductKey
ProductAlternateKey
ProductSubcategor...
WeightUnitMeasure...
SizeUnitMeasureC...
EnglishProductName
SpanishProductName
FrenchProductName
StandardCost
FinishedGoodsFlag
DimReseller
ResellerKey
GeographyKey
ResellerAlterna...
Phone
BusinessType
ResellerName
NumberEmplo...
OrderFrequency
OrderMonth
FirstOrderYear
DimCurrency
CurrencyKey
CurrencyAlternate...
CurrencyName
DimCustomer
CustomerKey
GeographyKey
CustomerAlternat...
Title
FirstName
MiddleName
LastName
NameStyle
BirthDate
MaritalStatus

Figure 4.15



A new column is added to the Employee table, as shown in Figure 4.16. The data type of this calculated column will be determined based on the data types of the actual columns involved in the calculation or data used within the expression. If the expression results in a number, the data type for this column is an integer. In the preceding example the data type of this column is a string.

Figure 4.16



The DSV maintains the named calculation definition in the metadata of the Analysis Services database; it does not write it out to the underlying tables. When you want to view the data of this table (which you see later in this chapter), the expression is added to the SQL query that Analysis Services sends to the relational backend. You can see the data in the computed column in the results that are returned.

Data Source Views in Depth

Data warehouse designs consist of one or more fact tables and associated dimension tables. Small data warehouses are usually composed of 10 to 20 tables, whereas larger data warehouses can have more than a hundred tables. Even though you have a large number of tables in your data warehouse, you likely work with a small subset of those tables at a time; each of which has relationships to other tables in the group. For example, assume you have sales, inventory, and human resources (HR) data to analyze and the HR data is not strongly related to the sales and inventory data but there is a desired linkage. Further, there may be one group of your customers who want to see just the HR-related view of the data and another group that wants to focus on just the sales and inventory information. You may even have a business policy that restricts the details of the HR information to a subset of your customers. To conform to these requirements, you might create two cubes, one for sales and inventory information and another one for HR. It is quite possible you can store the Sales, Inventory, and HR information in a single data source — in the ODS or OLTP system.

Employee HR information could be related to the sales and inventory information within the company in so far as there is a link between a given sales event and the employee who made the sale. You might want to slice the sales data by a specific employee, but to do so you must access information that is a part of a separate table only accessible to the HR department (for security reasons). You can get around this problem by making a single DSV containing all the tables that store sales, inventory, and HR information of a company. From that DSV, you can formulate both cubes and set permissions such that only members of the HR group can drill down on personal employee data.

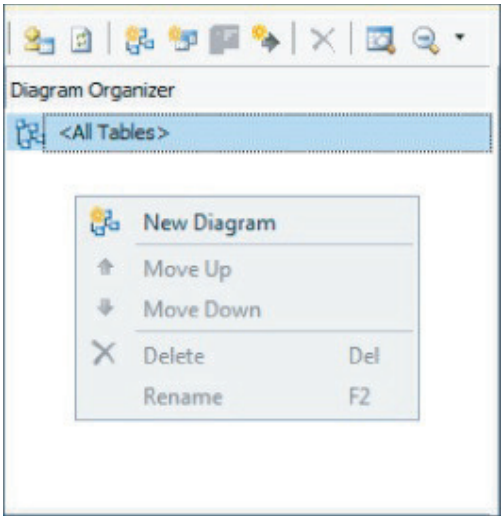
Diagrams

Having a lot of tables in the DSV definitely makes the navigation and usability a bit complex. When you work on HR data, you want to see only the tables related to this alone. For easy manageability you need customizable views within your DSV that show only certain tables. Analysis Services provides you with the ability to have several views within the DSV that each contains a subset of the DSV's tables. These views are called *diagrams*. By default, you get a diagram called <All Tables> when you complete the DSV Wizard that contains all the tables available in the DSV. You can create additional diagrams and choose the tables that you want to include within them. Next, you learn how to create a new diagram.

To create a new diagram, do the following:

1. Right-click the Diagram Organizer pane, and select New Diagram, as shown in Figure 4.17.

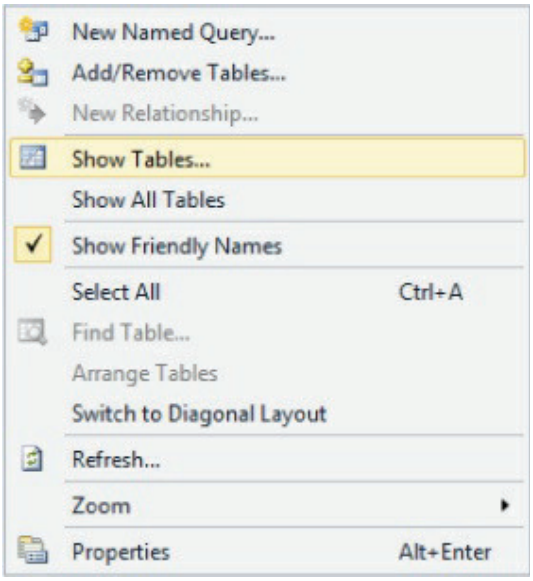
Figure 4.17



2. Name the new diagram Internet Sales.

3. You now have an empty diagram view. Right-click the Diagram pane and select Show Tables (see Figure 4.18). A dialog displays where you can choose the table(s) you want to include in this diagram.

Figure 4.18

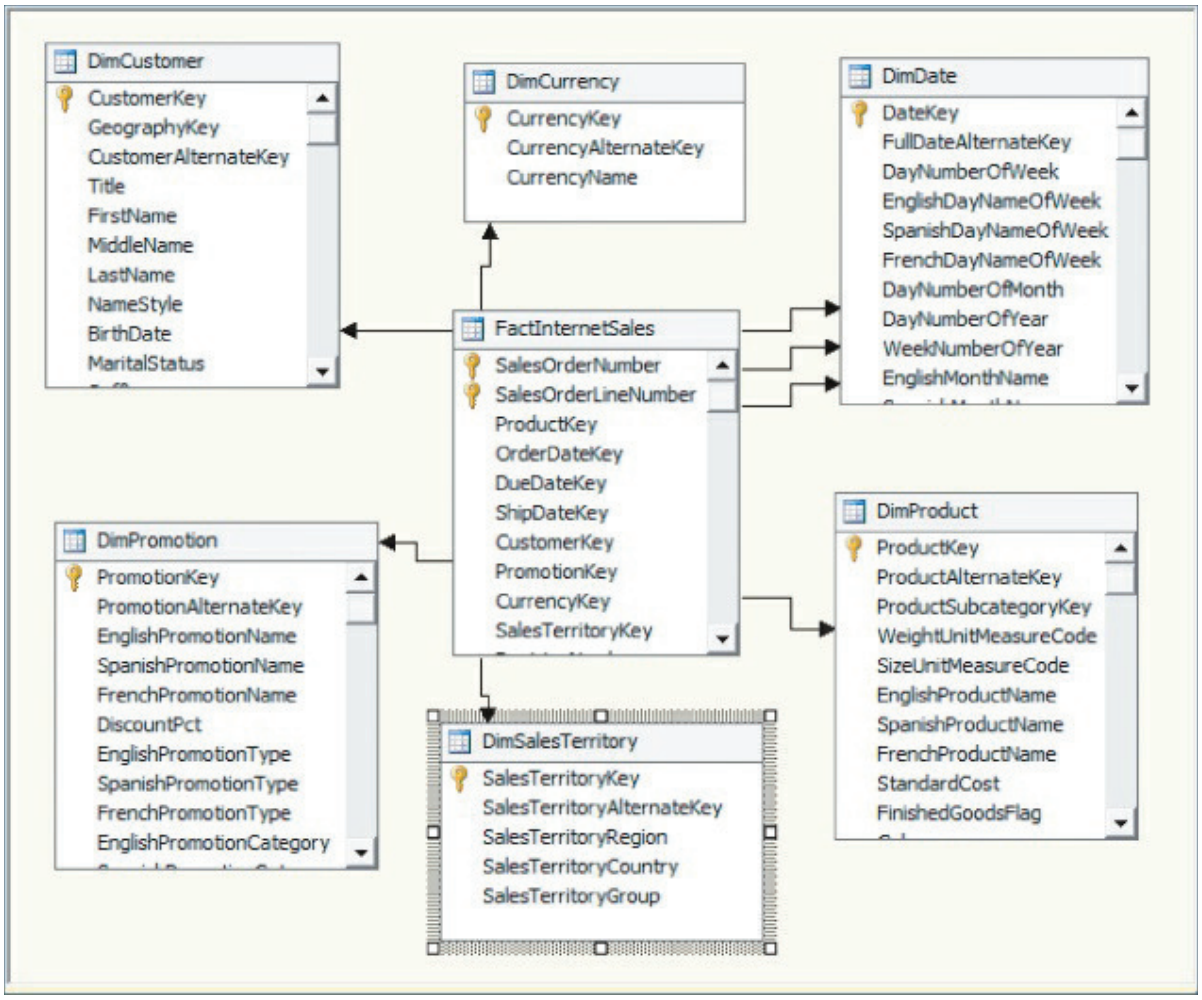


4. Select the FactInternetSales table, and click OK.

5. Right-click the caption of the table you just added, and select Show Related Tables.

This gives you a diagram that contains the FactInternetSales fact table and the related dimension tables, as shown in Figure 4.19. The Internet Sales diagram you have created has seven of the ten tables in the DSV. This diagram allows you to work with only the tables that related to Internet sales.

Figure 4.19



If you do not want a specific table in your diagram, you can right-click the table and select Hide. Instead of Steps 3 through 5 in the preceding list, you could have added tables to the diagram view by dragging and dropping them from the Table pane to the Diagram pane. Create another diagram called Reseller Sales and add the FactResellerSales table and related tables.

Data Source View Properties

Each object in your Analysis Services project has certain properties. Within the DSV Designer you can view the properties of the objects in the DSV such as tables, views, columns, and relationships. Properties of these objects are shown in the Properties window within SSDT, as shown in Figure 4.20.

Figure 4.20

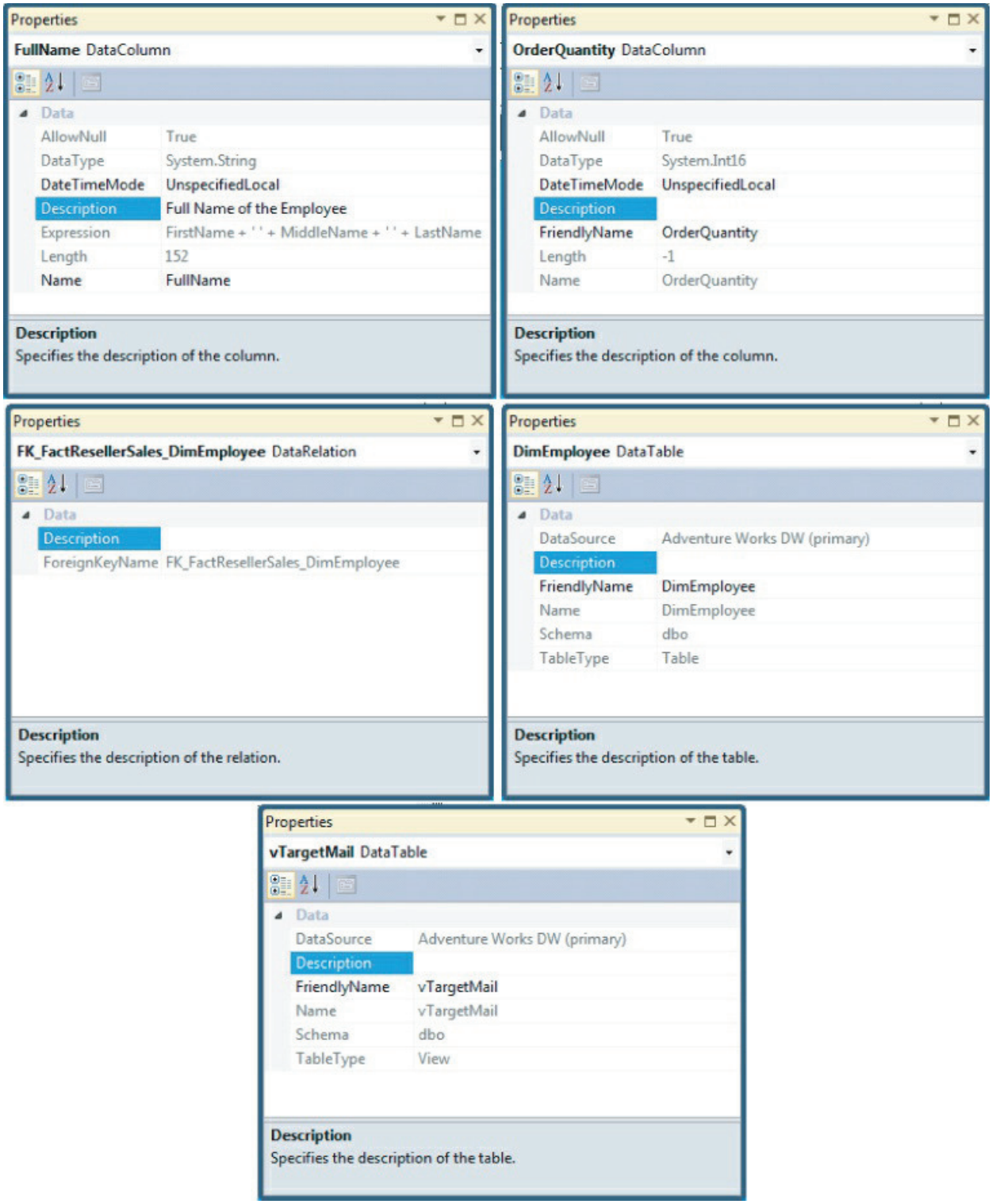


Figure 4.20 shows the properties of various DSV objects: a column in a table, a named calculation, a table, and a relationship. For regular columns in a table, you have the properties AllowNull, DataType, DateTimeMode, Description, FriendlyName, Length, and Name. SSDT populates the properties of a column by retrieving the corresponding properties from the data source. Based on the properties defined in the data source, the properties AllowNull, DataType, Length, Name, and FriendlyName are populated. The Length property is applicable only for the string data type. For all other data types, the Length property has a value of −1. You cannot change certain properties of a column. They are not editable in the Properties window and are grayed out to indicate they are read-only.

You can change the FriendlyName property and also provide a description for each column. Often, columns of a table in the relational database might not have user-friendly names. User-friendly means the name of the column should indicate clearly or intuitively the data stored in the column. The FriendlyName property is initially populated by the DSV Wizard with the name of the column in the data source. You can change this property to a name that is easier to understand and more indicative of the purpose of the data in the column. The DSV Designer provides you with the option of switching between the original column names and the friendly names. You can right-

click in the DSV diagram view and toggle between the friendly name and the original column name by selecting the Show FriendlyName option.

Named calculation columns do not have a FriendlyName property because you define the name of this column and should provide an intuitive and understandable name. Instead, named calculation columns have the Expression property because each named column is created from a SQL expression. You can change this expression only in the Named Column dialog and not in the Properties window.

Tables have the properties DataSource, Description, FriendlyName, Name, Schema, and TableType. The DataSource property is the name of the data source of the table. The TableType property indicates whether the object in the underlying data source is a table or a view. Similar to the columns, tables also have the option to specify a friendly name.

Relationships between tables are given a name that includes the tables that participate in the relationship. This name is also listed as the ForeignKeyName property. They do not have a FriendlyName property but they do allow you to add a description that will be included in the metadata of the DSV.

Different Layouts in DSVs

The DSV Designer provides two different layout types: rectangular and diagonal. In rectangular layout, the lines representing the relationships between tables are composed of horizontal and vertical lines, and the lines emerge from any of the four sides of the table. The second layout type offered by the DSV Designer is diagonal layout. In diagonal layout, lines emerge from only the right or left sides of the table and the lines themselves are not restricted to be only horizontal and vertical lines; they can be diagonal. You can switch between rectangular layout and diagonal layout in the DSV by right-clicking in the DSV Designer and selecting the layout type of your choice. Figures 4.21 and 4.22 show rectangular and diagonal layouts, respectively, of the Internet Sales diagram.

Diagonal layout works best for subset diagrams that contain a single fact table with its associated dimensions — a classic star or snowflake schema. For more complex diagrams, though, a proliferation of diagonal lines can make the diagram confusing. In that case, rectangular layout can be easier to view and understand. For this reason, the default layout type for the default diagram created by the DSV Wizard is rectangular.

Validating Your DSV and Initial Data Analysis

The relationships specified in the DSV are an important component of the structure of the dimensions and cubes in your multidimensional model. Therefore, validating your DSV is crucial to your data warehouse design. The DSV Designer provides a first level of validation when you specify relationships. If the data types of column(s) involved in the relationship do not match, the DSV will not allow you to establish the relationship. This forces you to make sure you appropriately cast the data types of the column(s) involved in the relationships. You might need another level of validation by looking at the data within each table. You can do this by issuing queries to the tables in the relational data source. The DSV provides a way to look at sample data for validation. A couple of validations you can do within the DSV by looking at sample data follow:

- Looking at the fact table data helps you make sure the table contains fact data, the primary key has been specified correctly, and appropriate relationships needed for dimensions are established.
- Analyzing a dimension table's sample data ensures that you have all the relationships established between the fact and dimension table and any relationships within each table are correctly established. For example, if you have an Employee table that contains an employee and his manager, you might want to establish a relationship so that your model can take advantage of this.

Looking at samples of data from the tables in the DSV helps you identify the measures of the cube as well as the hierarchies in each dimension. Analyzing sample data in the DSV also helps you identify dimensions that can be created from the fact table data. The analysis of sample data within the DSV is even more important when creating Data Mining models. You learn more about using Data Mining to analyze data in Chapter 12.

To see a sample of the data specified by your DSV, right-click a table in the DSV Designer, and select Explore Data. You can see rows from the underlying table presented within the Explore <tablename> Table window, as shown in Figure 4.23. The data presented is only a subset of the underlying table data. By default, the first 5,000 rows are retrieved and shown in this window. You can change the number of rows retrieved by clicking the Sampling Options button. Clicking the Sampling Options button launches the Data Exploration Options dialog where you can change the sampling method and the sample count. After you change the sample count value, you can click the Resample Data button to refresh the data based on the new settings. The Table tab shows the raw sampled data from the data source as rows and columns with column headings.

Figure 4.21

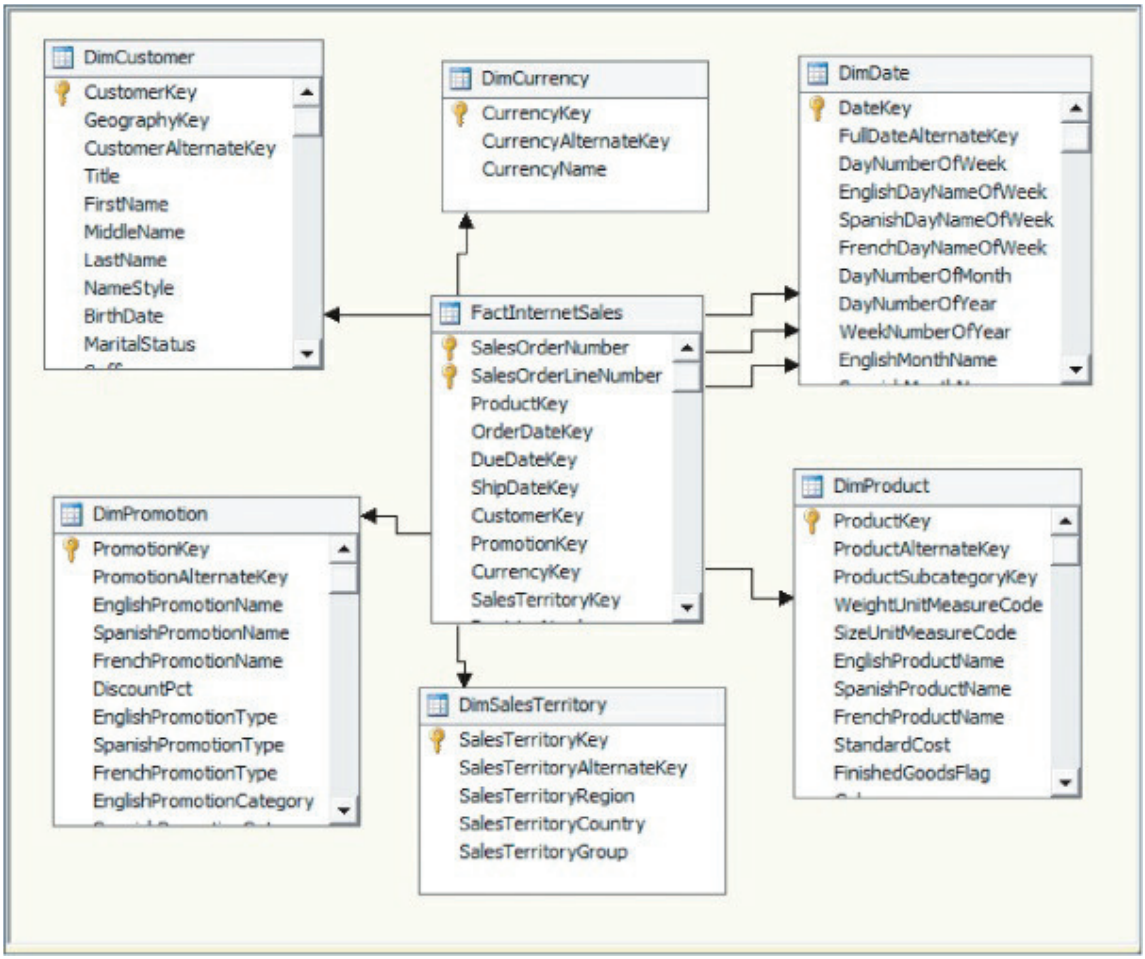

DimCustomer
CustomerKey
GeographyKey
CustomerAlternateKey
Title
FirstName
MiddleName
LastName
NameStyle
BirthDate
MaritalStatus
DimCurrency
CurrencyKey
CurrencyAlternateKey
CurrencyName
DimDate
DateKey
FullDateAlternateKey
DayNumberOfWeek
EnglishDayNameOfWeek
SpanishDayNameOfWeek
FrenchDayNameOfWeek
DayNumberOfMonth
DayNumberOfYear
WeekNumberOfYear
EnglishMonthName
FactInternetSales
SalesOrderNumber
SalesOrderLineNumber
ProductKey
OrderDateKey
DueDateKey
ShipDateKey
CustomerKey
PromotionKey
CurrencyKey
SalesTerritoryKey
DimPromotion
PromotionKey
PromotionAlternateKey
EnglishPromotionName
SpanishPromotionName
FrenchPromotionName
DiscountPct
EnglishPromotionType
SpanishPromotionType
FrenchPromotionType
EnglishPromotionCategory
DimProduct
ProductKey
ProductAlternateKey
ProductSubcategoryKey
WeightUnitMeasureCode
SizeUnitMeasureCode
EnglishProductName
SpanishProductName
FrenchProductName
StandardCost
FinishedGoodsFlag
DimSalesTerritory
SalesTerritoryKey
SalesTerritoryAlternateKey
SalesTerritoryRegion
SalesTerritoryCountry
SalesTerritoryGroup

Figure 4.22

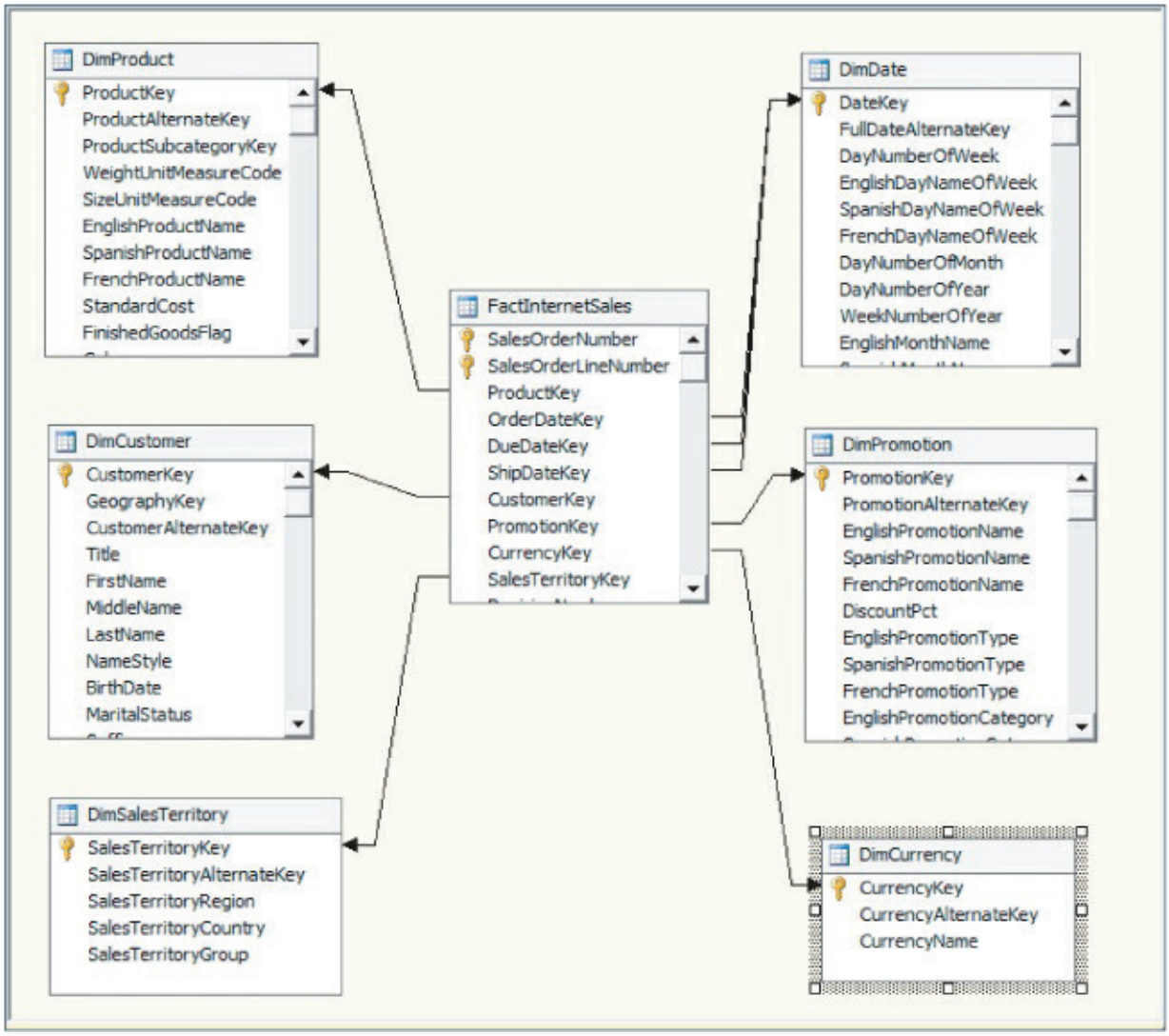
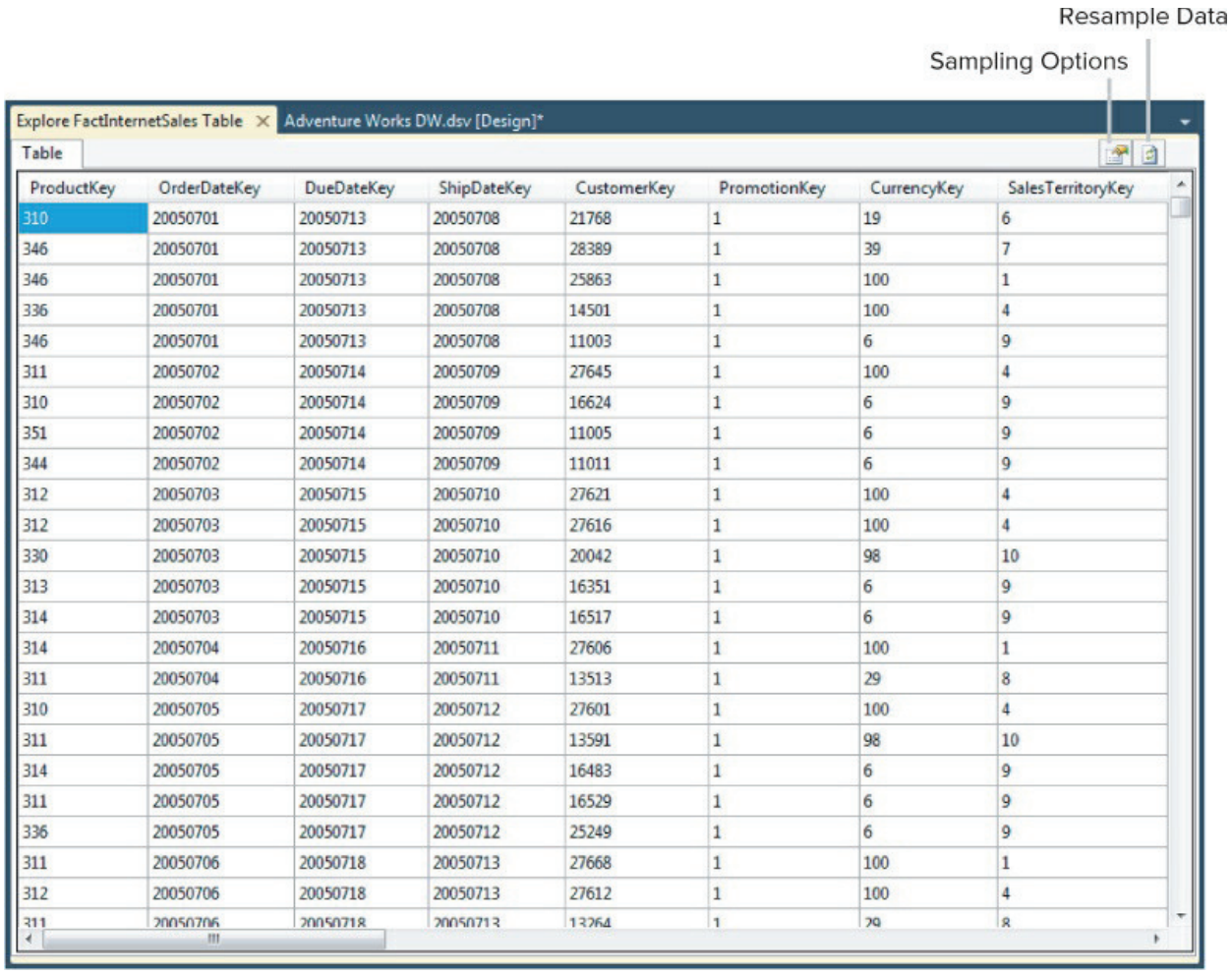

DimProduct
ProductKey
ProductAlternateKey
ProductSubcategoryKey
WeightUnitMeasureCode
SizeUnitMeasureCode
EnglishProductName
SpanishProductName
FrenchProductName
StandardCost
FinishedGoodsFlag
DimDate
DateKey
FullDateAlternateKey
DayNumberOfWeek
EnglishDayNameOfWeek
SpanishDayNameOfWeek
FrenchDayNameOfWeek
DayNumberOfMonth
DayNumberOfYear
WeekNumberOfYear
EnglishMonthName
FactInternetSales
SalesOrderNumber
SalesOrderLineNumber
ProductKey
OrderDateKey
DueDateKey
ShipDateKey
CustomerKey
PromotionKey
CurrencyKey
SalesTerritoryKey
DimCustomer
CustomerKey
GeographyKey
CustomerAlternateKey
Title
FirstName
MiddleName
LastName
NameStyle
BirthDate
MaritalStatus
DimPromotion
PromotionKey
PromotionAlternateKey
EnglishPromotionName
SpanishPromotionName
FrenchPromotionName
DiscountPct
EnglishPromotionType
SpanishPromotionType
FrenchPromotionType
EnglishPromotionCategory
DimSalesTerritory
SalesTerritoryKey
SalesTerritoryAlternateKey
SalesTerritoryRegion
SalesTerritoryCountry
SalesTerritoryGroup
DimCurrency
CurrencyKey
CurrencyAlternateKey
CurrencyName

Figure 4.23



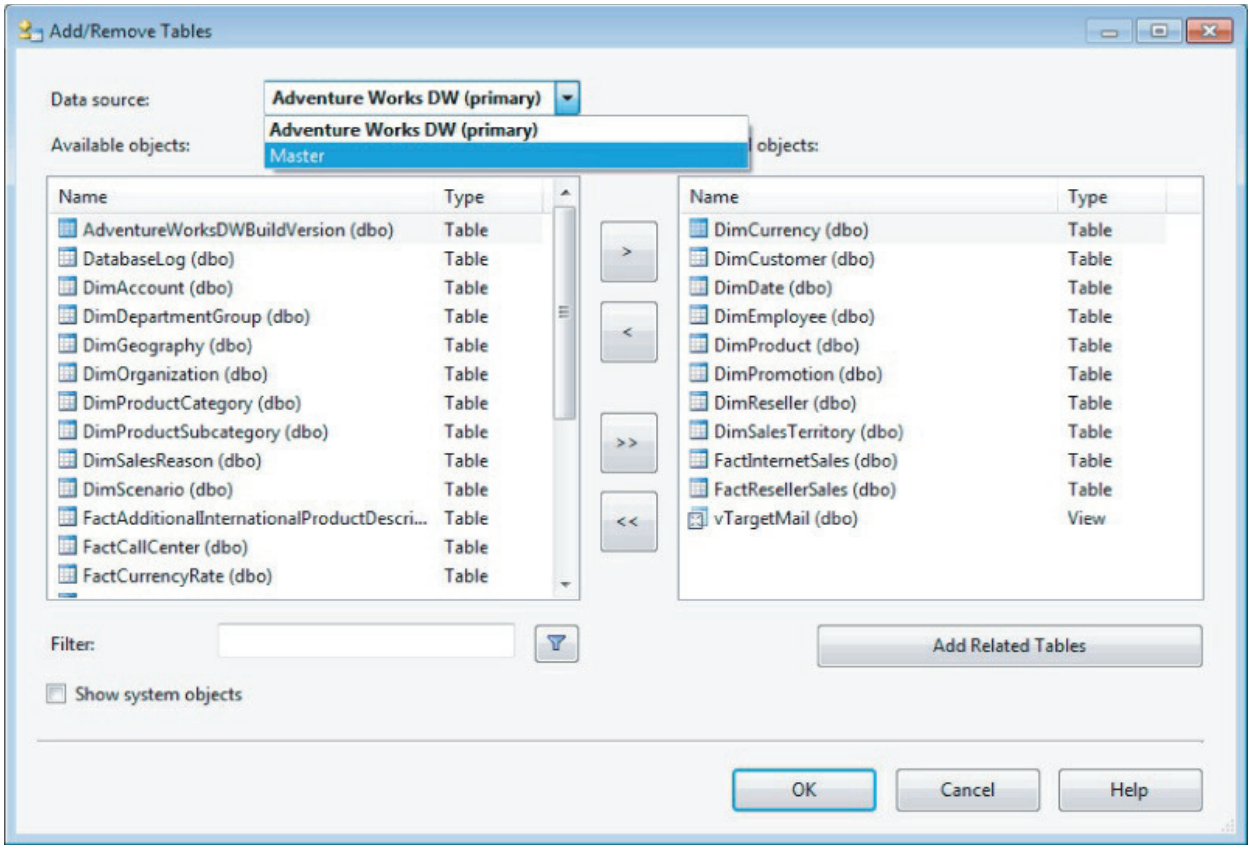
ProductKey	OrderDateKey	DueDateKey	ShipDateKey	CustomerKey	PromotionKey	CurrencyKey	SalesTerritoryKey
310	20050701	20050713	20050708	21768	1	19	6
346	20050701	20050713	20050708	28389	1	39	7
346	20050701	20050713	20050708	25863	1	100	1
336	20050701	20050713	20050708	14501	1	100	4
346	20050701	20050713	20050708	11003	1	6	9
311	20050702	20050714	20050709	27645	1	100	4
310	20050702	20050714	20050709	16624	1	6	9
351	20050702	20050714	20050709	11005	1	6	9
344	20050702	20050714	20050709	11011	1	6	9
312	20050703	20050715	20050710	27621	1	100	4
312	20050703	20050715	20050710	27616	1	100	4
330	20050703	20050715	20050710	20042	1	98	10
313	20050703	20050715	20050710	16351	1	6	9
314	20050703	20050715	20050710	16517	1	6	9
314	20050704	20050716	20050711	27606	1	100	1
311	20050704	20050716	20050711	13513	1	29	8
310	20050705	20050717	20050712	27601	1	100	4
311	20050705	20050717	20050712	13591	1	98	10
314	20050705	20050717	20050712	16483	1	6	9
311	20050705	20050717	20050712	16529	1	6	9
336	20050705	20050717	20050712	25249	1	6	9
311	20050706	20050718	20050713	27668	1	100	1
312	20050706	20050718	20050713	27612	1	100	4
311	20050706	20050718	20050713	13264	1	29	8

Multiple Data Sources Within a DSV

Data warehouses can consist of data from several data sources. Some examples of data sources are SQL Server, Oracle, DB2, and Teradata. Traditionally, the OLTP data is transferred from the operational data store to the data warehouse — the staging area that combines data from disparate data sources. This is not only time-intensive in terms of design, maintainability, and storage but also in terms of other considerations such as replication of data and ensuring data is in sync with the source. Analysis Services helps you avoid this and gives you a better return on your investment.

The DSV Designer has the ability to include tables from multiple data sources in a single DSV, which you can then use to build your cubes and dimensions. You first need to define the data sources that include the tables that are part of your data warehouse design using the Data Source Wizard. When this has been accomplished, you create a DSV and include tables from one of the data sources. This first data source is called the primary data source and needs to be a SQL Server data source. You can then add tables and work with them in the DSV Designer as described earlier in this chapter. The Add/Remove Tables dialog allows you to choose a different data source, as shown in Figure 4.24 so that you can add its tables to the DSV. To illustrate the selection of tables from a second data source, as shown in Figure 4.24, a second data source in the AnalysisServicesMultidimensionalTutorial project was created from the SQL Server Master database. You should be aware that there might be performance implications when retrieving data from secondary data sources because all the queries are routed through the primary SQL Server data source.

Figure 4.24



After you add the tables from multiple data sources to your DSV, you can start creating your cubes and dimensions as if these came from a single data source. The limitation that the primary data source needs to be a SQL Server is due to the fact that Analysis Services instance uses a SQL Server-specific feature called OPENROWSET to retrieve data from other data sources.

Summary

You now have the skills to deal with the challenges real-world data warehouses can throw at you when using multiple data sources. You learned about OLE DB and managed data providers used by Analysis Services to retrieve data from data sources and the trade-offs of using one versus another. Indeed, you learned that you can tame the disparate data source beast by using multiple data sources. Then you learned to consolidate the tables and relationships of interest in Data Source Views (DSVs), and finally, to refine the tables and relationships in the DSVs so you have to deal with only what's relevant.

When you make changes in the DSV, that is where the changes stay — in the DSV. The changes are not written out to the underlying tables as you might expect. This is a good thing. To see why, take a look at the alternative to using the DSV capability. The alternative method is to create a view in SQL with real relationship transforms in the underlying tables. That's another viable approach, but if your data spans multiple databases, you may have to create linked servers and that can become time-consuming. Analysis Services provides an easy way to specify these cross-database relationships within a DSV without the overhead of having to use linked servers. However, when multiple data sources are included in a single DSV, the primary data source must support the ability to send queries and retrieve results from other servers. You can incur performance degradation due to this method; however, you do avoid the complexity of managing data on multiple servers to create your data warehouse.

You're doing great! You're now ready to tackle core business intelligence constructs such as dimension design (Chapter 5) and cube design (Chapter 6). If you already know these topics from working with earlier versions of SQL Server Analysis Services, work through the chapters anyway. There have been some important changes to the Cube and Dimension Wizards in Analysis Services 2012.

Chapter 5

Dimension Design

What's in this chapter?

- Creating a dimension using the Dimension Wizard
- Creating attributes
- Defining attribute relationships
- Creating user hierarchies
- Browsing dimensions
- Understanding properties of dimensions and attributes
- Defining dimension translations
- Creating a snowflake dimension
- Creating a Time dimension
- Creating a parent/child hierarchy

Consider airline travel. Many people with different ultimate destinations embark on a journey together. They board a single airplane at a common location. The airline takes all of them to a common destination. (For the purposes of this metaphor, assume a nonstop flight.) After they arrive at that destination, they branch out to where their own business or personal agendas take them.

The data warehouse design process using Analysis Services multidimensional development tools is similar to that. The wizards that create each major object type take everyone to a common place. When you get there, you can use the corresponding designer to take your objects to their ultimate destination. You saw this approach in the previous chapter when you learned about creating data sources and Data Source Views. Likewise, in this chapter you learn how to create dimensions in a common way using the Dimension Wizard; then you use the Dimension Designer to take your dimensions to their final destination based on your business needs.

Cubes are made up of dimensions and measures, where the measure values are aggregated along each dimension. Without an understanding of dimensions and how measures are aggregated along them, you can't create and exploit the power of cubes, so jump right into learning about building and viewing dimensions. After you create dimensions, you add them to a cube and define the right relationship types between the fact data and dimensions. Analysis Services supports six relationship types: no relationship, regular, fact, referenced, many-to-many, and data mining. You learn about these relationship types in this chapter and in Chapters 8 and 12. In addition, you learn about the attributes and hierarchies that are an integral part of dimensions. You learn how to model Time dimensions and Parent-Child dimensions, which are different from regular dimensions and are important components of many data warehouses. Finally, you learn how to process and browse dimensions.

Working with the Dimension Wizard

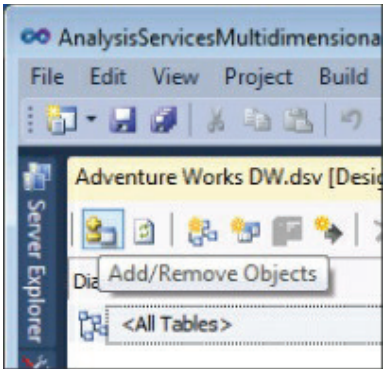
Dimensions help you define the structure of your cube to facilitate effective data analysis. Specifically, dimensions provide you with the ability to slice a cube's data. You can build dimensions from one or more dimension tables. As you learned in Chapter 1, you can design your data warehouse using star or snowflake schemas. In a star schema, you create dimensions from single tables joined to a fact table. In a snowflake schema, you create dimensions from two or more joined dimension tables. When you build your cube using snowflake dimensions, one of the tables that make up the dimension joins to the fact table. You create both of these dimension types in this chapter.

You also learned in Chapters 2 and 3 that each dimension contains objects called *hierarchies*. In Analysis Services you have two types of hierarchies to contend with: attribute hierarchies, which correspond to a single column in a relational table, and multilevel or user hierarchies, which derive from two or more attribute hierarchies. Each level in the multilevel hierarchy is an attribute hierarchy. A typical example of an attribute hierarchy would be a ZIP code in a Geography dimension, and a typical example of a user hierarchy would be Country-State-City-ZIP code, also in a Geography dimension. In everyday discussions of user hierarchies, most people leave off the "user" and just call them "hierarchies."

For the exercises in this chapter, you start from the project you created in Chapter 2. If you don't have that project handy, you can download it from www.wrox.com. Regardless of whether you download it or use the project you built in Chapter 2, you need to add the Geography dimension (dbo.DimGeography) to the DSV. To add this dimension to your DSV, follow these steps:

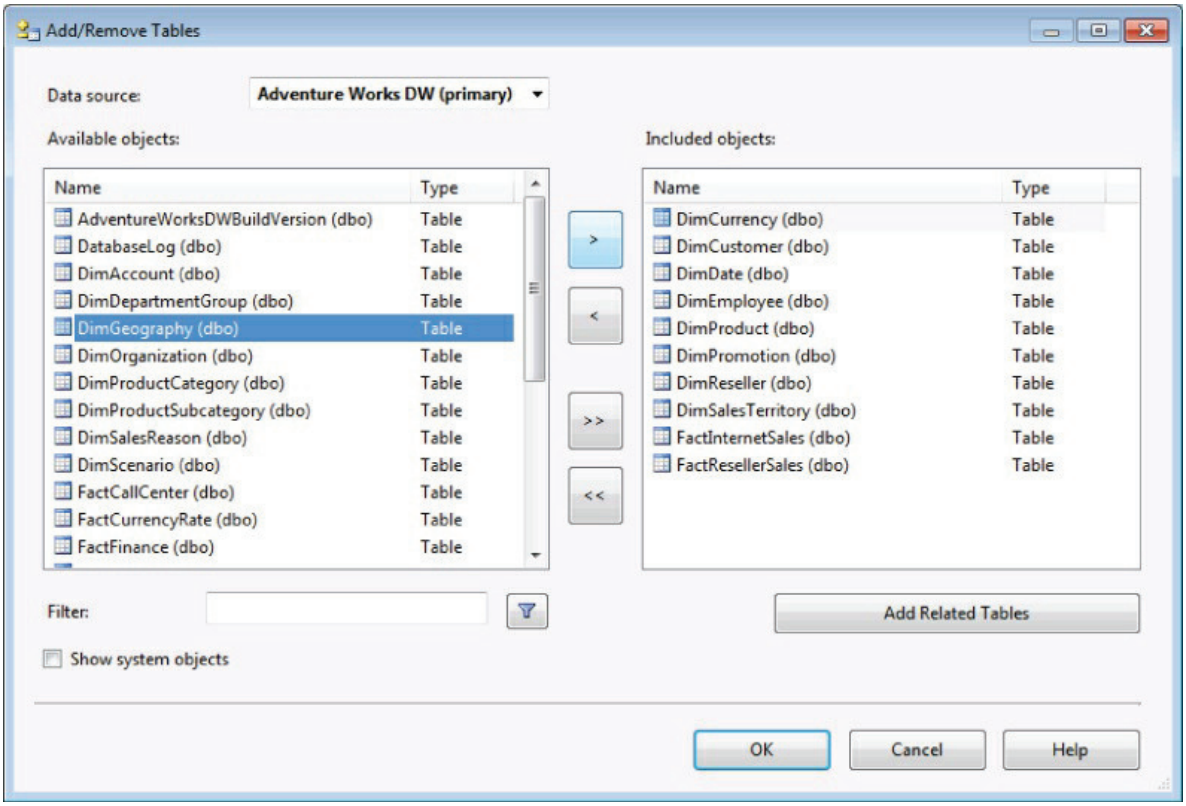
1. In the Solution Explorer, double-click the Adventure Works DW.dsv DSV.
2. Click the Add/Remove Objects toolbar button (the top-left button in the DSV Designer toolbar) as shown in Figure 5.1.

Figure 5.1



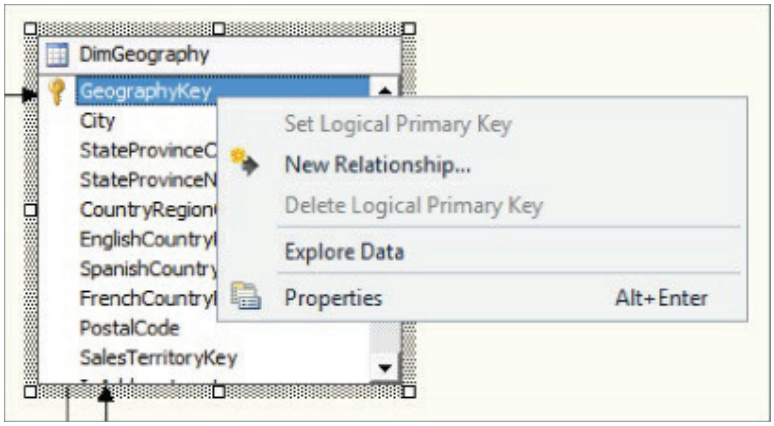
3. In the Available objects list, select DimGeography, and click the > (right arrow) button, as shown in Figure 5.2. This moves the DimGeography dimension table into the Included objects list in the Add/Remove Tables dialog. Click OK to continue.

Figure 5.2



4. All dimension tables in a DSV should have a primary key set to allow Analysis Services to determine the key attribute column of the dimension, which will be joined to a foreign key column in the appropriate fact table(s). Review the DimGeography table in the DSV Designer to see that the DimGeography table's GeographyKey column has been set as the primary key, as shown in Figure 5.3.

Figure 5.3

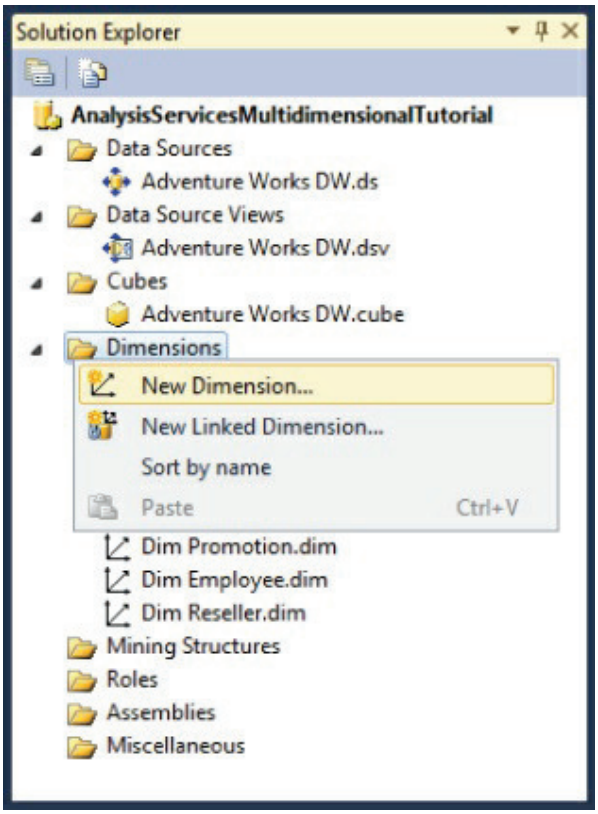


If a primary key is not defined in the source data table, you can right-click the desired key column(s) in that table and select Set Logical Primary Key (refer to Figure 5.3). This enables you to create the right primary key/foreign key relationships for tables in your data warehouse even if they are not defined in the source data. If the primary key for the table has been defined in the source data and recognized by the DSV Designer, the option to Set Logical Primary Key will be disabled.

Now you are ready to explore the use of the Dimension Wizard in SSDT. Continue to follow these steps to create the Geography dimension.

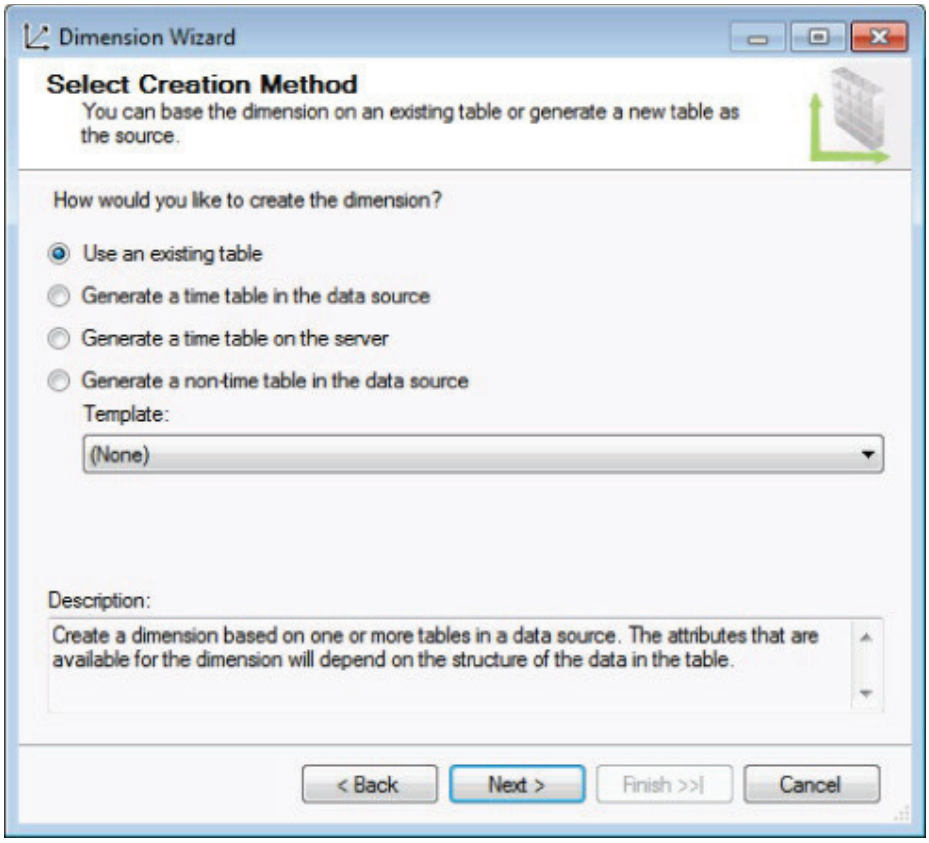
5. Launch the Dimension Wizard by right-clicking the Dimensions folder in the Solution Explorer and selecting New Dimension, as shown in Figure 5.4. If the welcome screen of the Dimension Wizard appears, click Next.

Figure 5.4



6. On the Select Creation Method page, as shown in Figure 5.5, you see four options:

Figure 5.5



- Use an existing table
- Generate a time table in the data source
- Generate a time table on the server
- Generate a non-time table in the data source

Using an existing table in the data source enables the creation of a standard dimension that can later be modified to become any sophisticated dimension type. This option is a great starting point for creating most dimensions.

A Time dimension, on the other hand, is a unique type of dimension typically created from a table that contains time information such as year, semester, quarter, month, week, and date. A Time dimension is unique because its members are fixed (for example, a year always has 12 months in it) and typical business analyses are performed over time. Due to the uniqueness of the Time dimensions and how they are used in business analysis, there are special MDX functions that can be used with them. Furthermore, aggregation of data on a Time dimension does not need to be a garden-variety additive aggregation like sum or count.

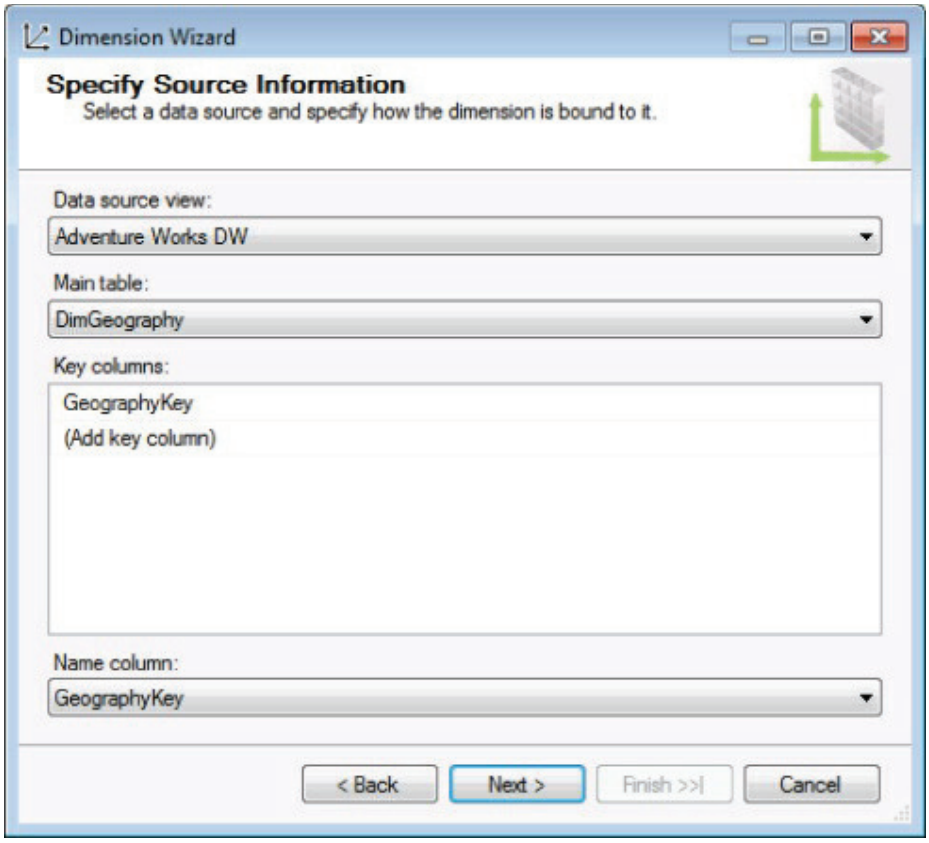
Most business decision makers want to analyze their data across a Time dimension to understand, for example, the month with maximum sales for a quarter or some other time period. Analysis Services provides you a distinct way to aggregate measure data across a Time dimension. This is done with semi-additive measures. You learn more about semi-additive measures in Chapters 6 and 9. In a Time dimension, some standard hierarchies are commonly used, such as fiscal year and calendar year, both of which can be built automatically. You can build the Time dimensions using either a table from the data source or without any associated tables in the data source. To create a table to serve as the source of your Time dimension, you choose the second option on the Select Creation Method page of the wizard. To create a server-based Time dimension, select the third option. You learn more about server-based Time dimensions in Chapter 8.

In addition to Time, Analysis Services is also aware of several other common dimension types used in business intelligence (BI) applications, such as Account, Customer, Employee, and Currency, and can create the necessary tables in the data source to support these dimension types. This chapter explores creating dimensions from existing tables in a data source.

In the Select Creation Method page, select Use an Existing Table, and click Next.

7. On the Specify Source Information page (shown in Figure 5.6), you need to select the DSV for creating the dimension, the main table from which the dimension is to be designed, the key columns for the dimension, and optionally a name column for the dimension key value. By default, the first DSV in your project is selected. Because the current project has only one DSV (the Adventure Works DW DSV), it is selected.

Figure 5.6



On the Main table drop-down on the page, you need to select the main table from which the dimension is to be designed. If a dimension is to be created from a star schema, the dimension is created from only one table. A dimension built using a snowflake schema contains several tables, one of which is the primary table of the dimension. This primary table is chosen as the main table in the Main table selection of the Dimension Wizard.

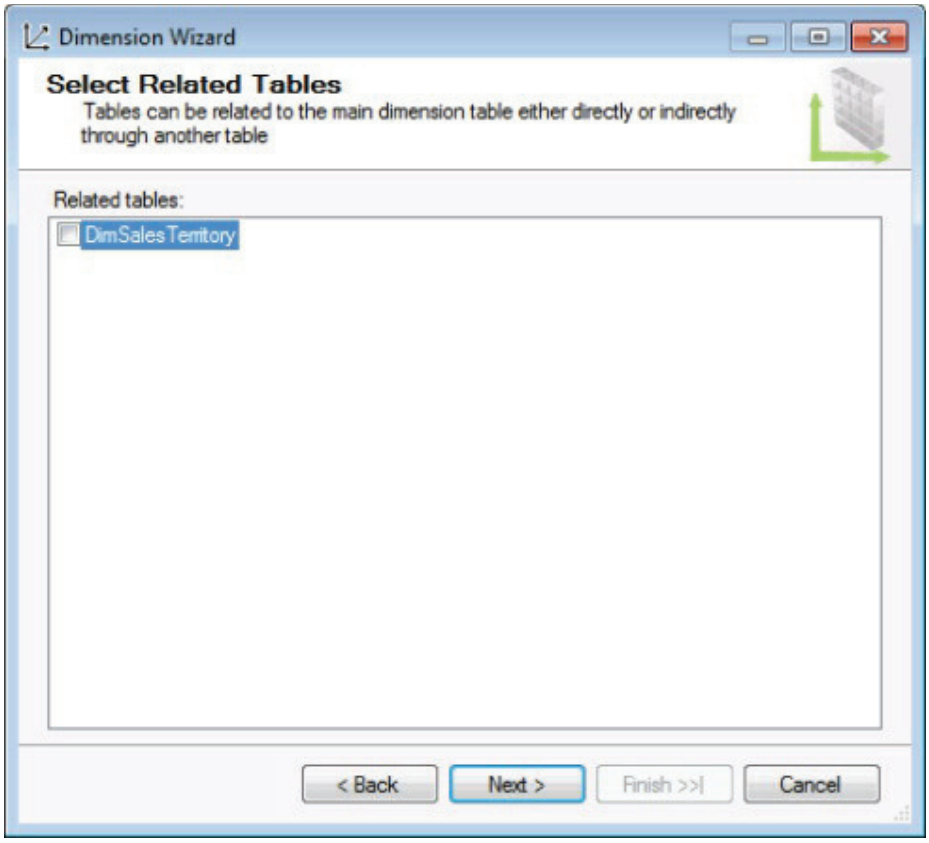
Select the DimGeography table from the Main table drop-down list, as shown in Figure 5.6.

After selecting the Main table for the dimension, the Key columns list and the Name column field are automatically set to the primary key of the selected table. If no primary key or logical primary key is set, you need to specify the key columns. If the primary key for the main dimension table is made up of multiple columns, you must select a single column in the Name column drop-down before proceeding. Because the DimGeography table has a primary key defined, the wizard used it to fill in the Key columns and Name column fields when you chose DimGeography as the Main table.

Click the Next button to proceed to the next step in the Dimension Wizard.

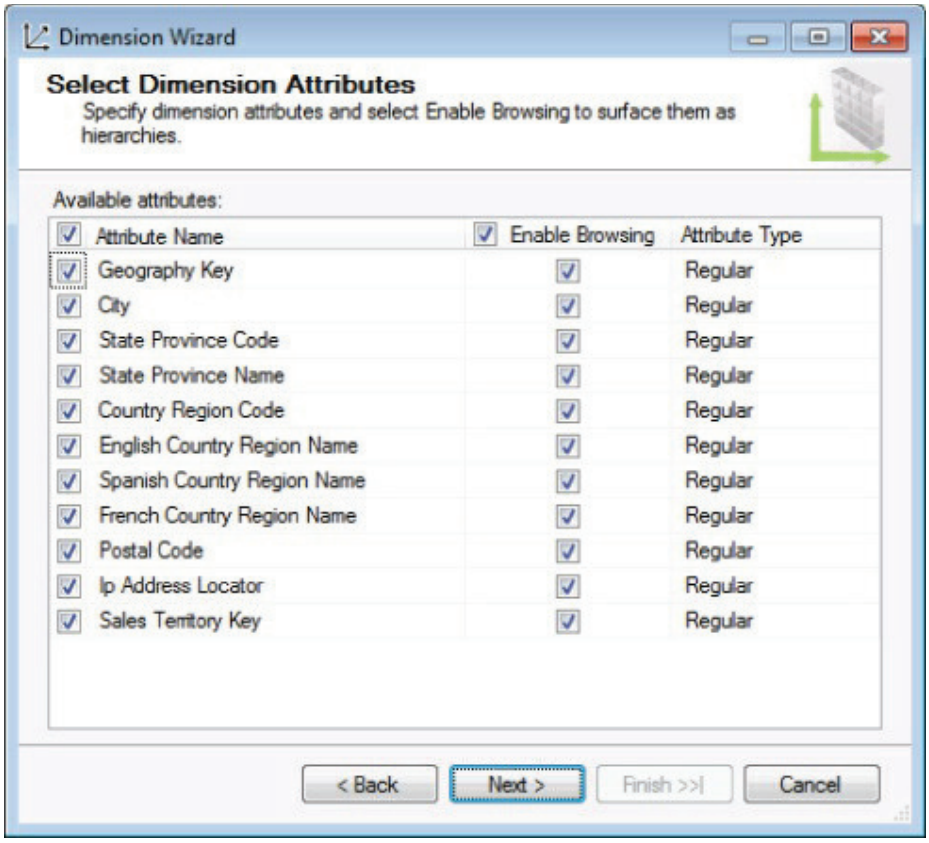
8. The Dimension Wizard now analyzes the DSV to detect any outward-facing relationships from the DimGeography table. An outward-facing relationship is a relationship between the DimGeography table and another table, such that a column in the DimGeography table is a foreign key related to another table. The Select Related Tables page (see Figure 5.7) shows that the wizard detected an outward relationship between the DimGeography table and the DimSalesTerritory table. In this example you model the DimGeography table as a star schema table instead of a snowflake schema. Deselect the DimSalesTerritory table and click Next.

Figure 5.7



9. The Select Dimension Attributes page of the Dimension Wizard (see Figure 5.8) displays the columns of the main table that have been selected for the dimension you're creating. The columns selected from the relational table are transformed by the dimension wizard into the attributes of the dimension that can then be used when querying the cube. You can control which of the attributes are available for browsing and querying by checking or unchecking the Enable Browsing option for each attribute. In addition, you can set the Attribute Type property to allow Analysis Services to provide special functionality based on the attribute's type. By default the wizard creates attribute names based on column names. You can change the attribute name for each selected attribute on this page.

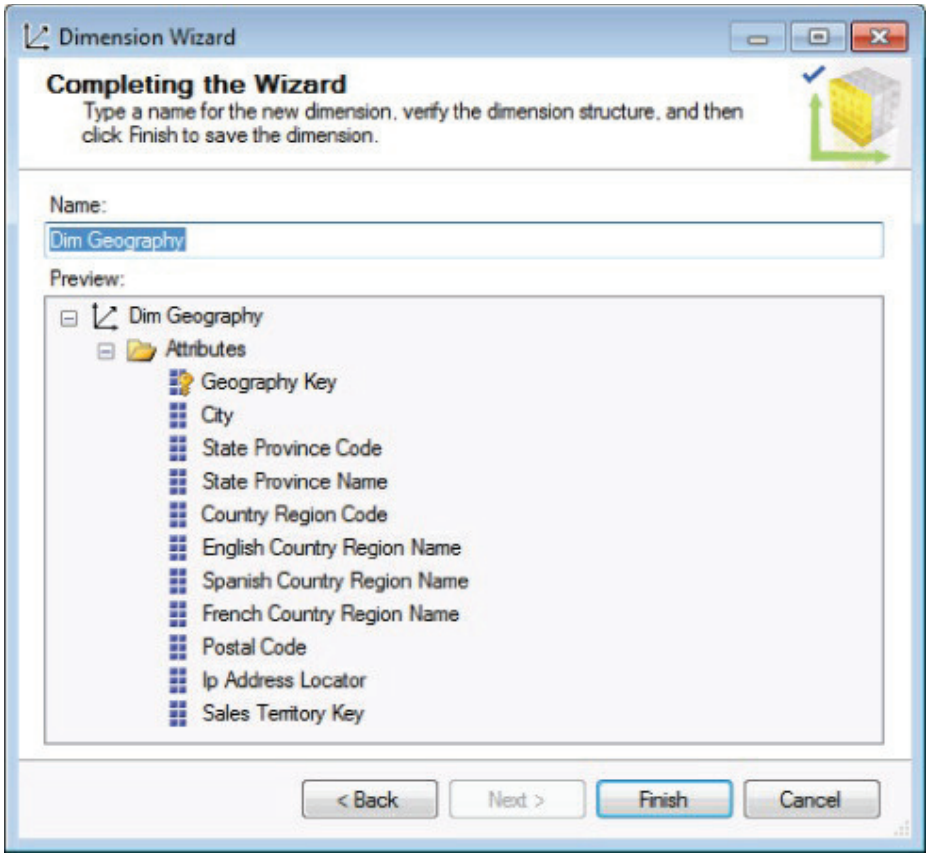
Figure 5.8



On the Select Dimension Attributes page, select all the columns of the DimGeography table. Leave their Attribute Type as Regular and allow them to be browsed, as shown in Figure 5.8. Click Next.

10. The final screen of the Dimension Wizard shows the attributes that will be created for the dimension based on your choices (see Figure 5.9). Click the Finish button.

Figure 5.9

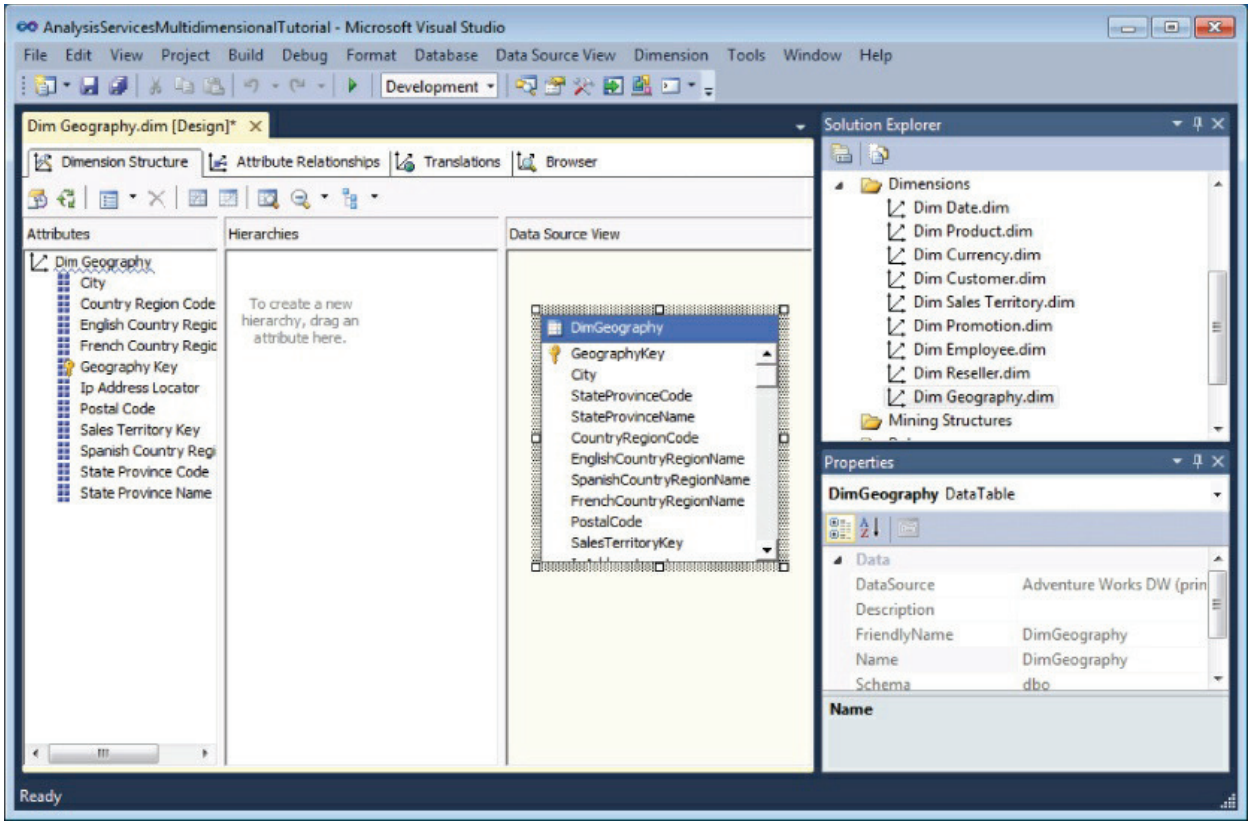


The wizard has created the Dim Geography dimension and opened it up in the Dimension Designer. Congratulations! You have successfully created your first dimension using the Dimension Wizard. Next, you learn how to use the Dimension Designer to enhance the dimension to fit your business needs.

Working with the Dimension Designer

The Dimension Designer, shown in Figure 5.10, is an important tool that helps you refine dimensions created by the Dimension Wizard. You can set properties such as unary operators, custom roll-ups, and so forth that help you define how data should be aggregated for cells referred to by members of hierarchies in the dimension. The Dimension Designer is composed of four pages, which you can access from the tabs at the top of the designer: Dimension Structure, Attribute Relationships, Translations, and Browser. The first of these pages, Dimension Structure, contains three panes: Attributes, Hierarchies, and Data Source View. The Attributes pane shows the dimension's attributes; the Hierarchies pane shows its hierarchies along with their levels; and the Data Source View pane shows the tables used in the dimension. In addition, you have a toolbar that contains buttons that help you enhance the dimension. If you hover over each toolbar button you can see a tooltip that describes the functionality of that button. Some of the buttons are the same as the ones you saw in the DSV Designer and are used for operations within the Dimension Designer's Data Source View pane. The functionality of the other buttons is discussed later in this chapter and in Chapter 8.

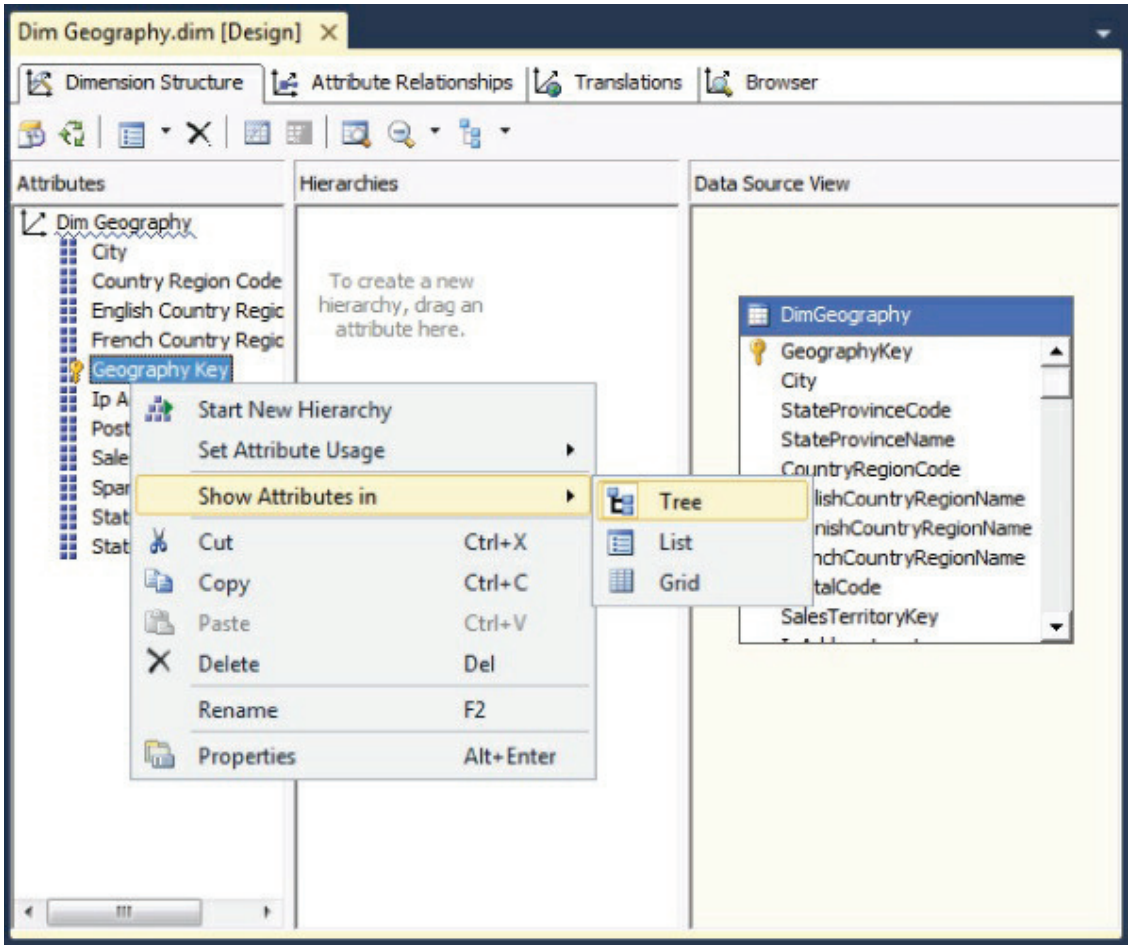
Figure 5.10



Attributes

Attribute hierarchies (or attributes, for short) are hierarchies that have only two levels: the leaf level, which contains one member for each distinct attribute value, and the All level, which contains the aggregated value of all the leaf level's members. The All level is optional. Each attribute directly corresponds to a table's column in the DSV. The Attributes pane in the Dimension Designer shows all the attribute hierarchies of the dimension. The default view in the Attributes pane is a Tree view, as shown in Figure 5.11. Two additional views are available: List view and Grid view. These views show the attributes and associated properties in different ways.

Figure 5.11



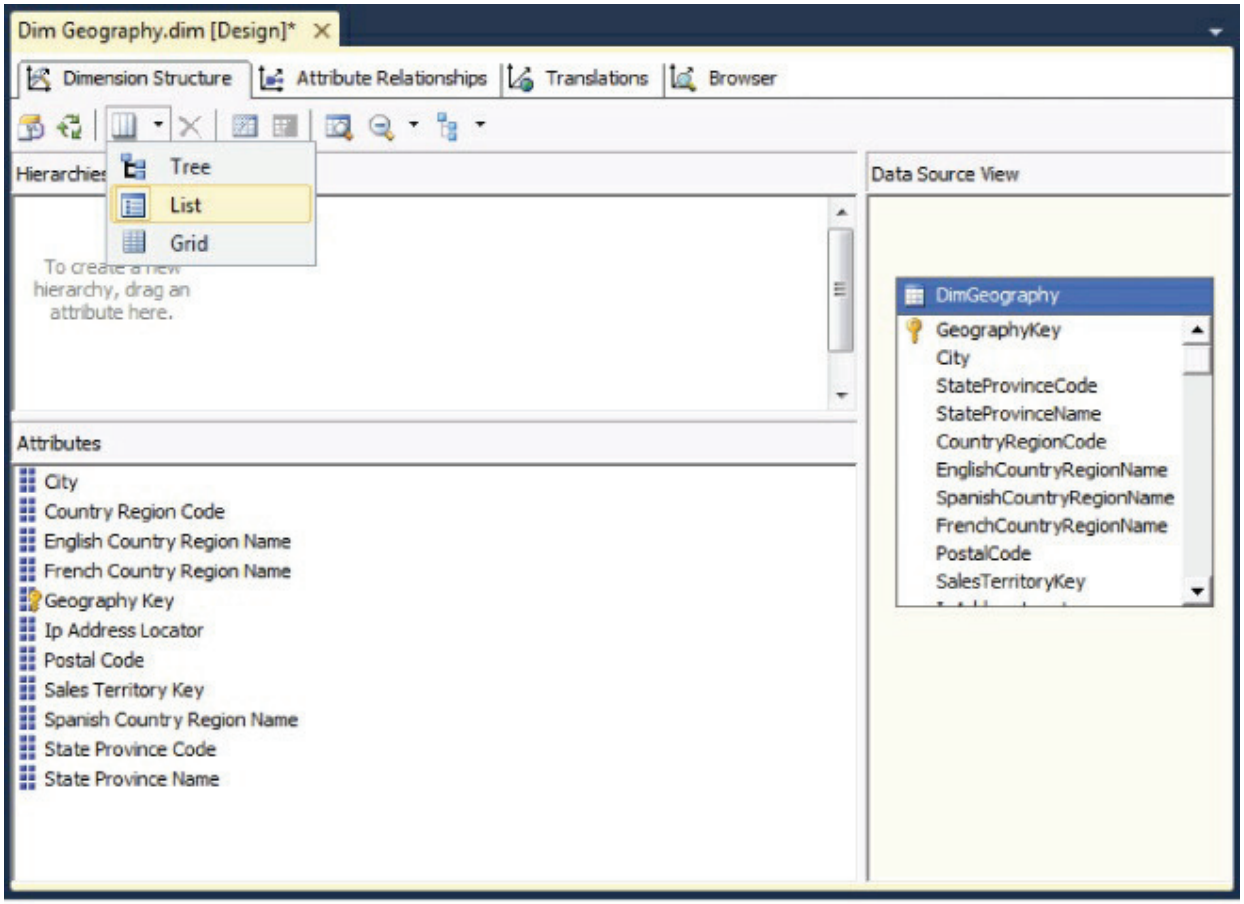
The List view repositions the Attributes pane below the Hierarchies pane and shows only the attributes of the dimension in a flat list. (It doesn't show the dimension name as a parent node.) This view is useful when your dimension has a lot of user hierarchies. Because you get a wider area for the Hierarchies pane, more hierarchies are visible without scrolling.

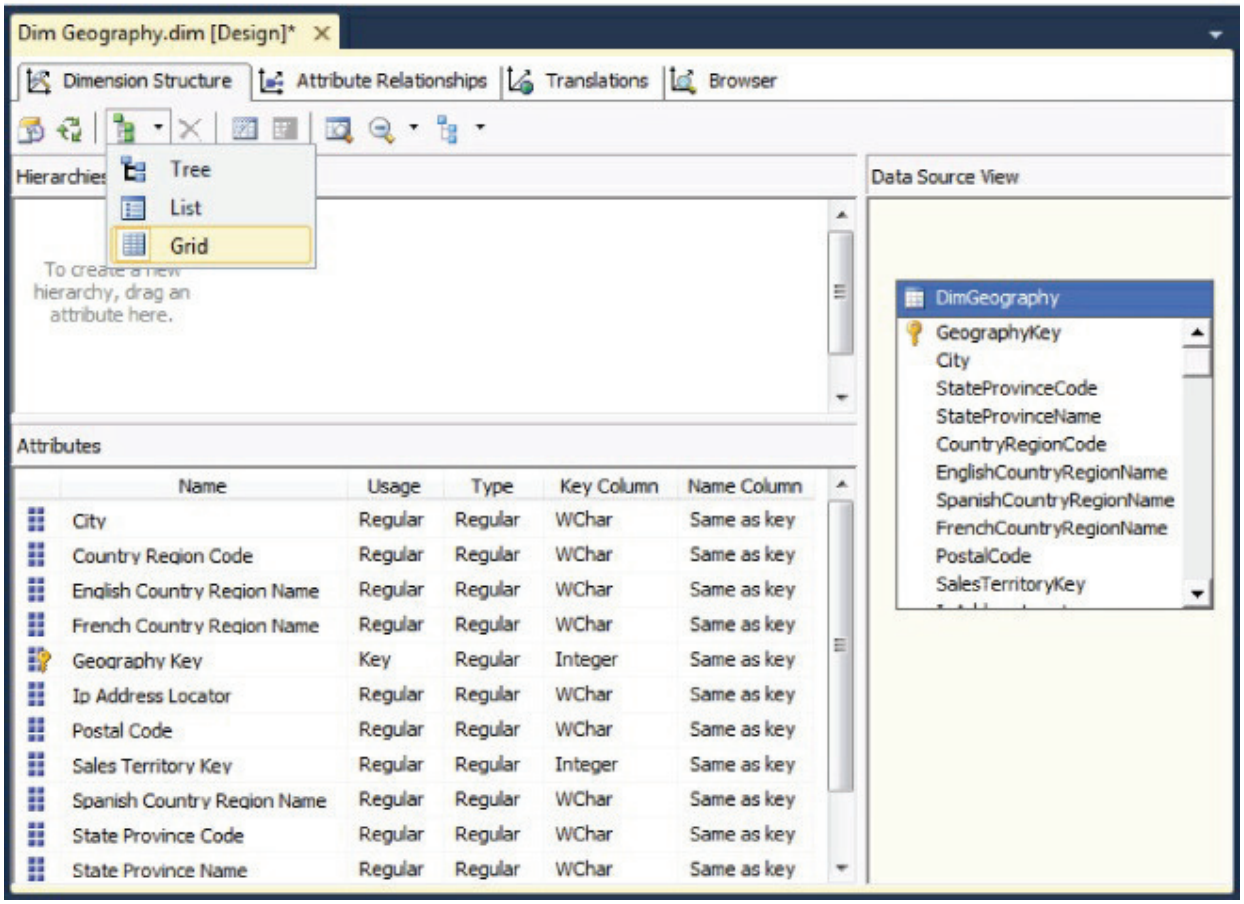
The Grid view is laid out similarly to the List view but includes additional columns that allow you to easily edit some of the important attribute properties right in the Attributes pane. When you work with more than one attribute, editing these properties in the Attributes pane is less cumbersome than having to select each attribute and then switching over to the Properties window to change the attribute's value. (All the properties shown in the Grid view are also present in the Properties window.)

You can toggle between the different views by right-clicking in the Attributes pane and selecting the view type you want, as shown in Figure 5.11. Choose the view that best helps you to visualize and design your dimension.

The List view and Grid view of the Dimension Designer are shown in Figure 5.12.

Figure 5.12


Dim Geography.dim [Design]*
Dimension Structure
Attribute Relationships
Translations
Browser
Tree
List
Grid
Hierarchies
To create a new hierarchy, drag an attribute here.
Data Source View
DimGeography
GeographyKey
City
StateProvinceCode
StateProvinceName
CountryRegionCode
EnglishCountryRegionName
SpanishCountryRegionName
FrenchCountryRegionName
PostalCode
SalesTerritoryKey
Attributes
City
Country Region Code
English Country Region Name
French Country Region Name
Geography Key
Ip Address Locator
Postal Code
Sales Territory Key
Spanish Country Region Name
State Province Code
State Province Name


Dim Geography.dim [Design]*
Dimension Structure
Attribute Relationships
Translations
Browser
Tree
List
Grid
Hierarchies
To create a new hierarchy, drag an attribute here.
Data Source View
DimGeography
GeographyKey
City
StateProvinceCode
StateProvinceName
CountryRegionCode
EnglishCountryRegionName
SpanishCountryRegionName
FrenchCountryRegionName
PostalCode
SalesTerritoryKey
Attributes
Name
Usage
Type
Key Column
Name Column
City Regular Regular WChar Same as key
Country Region Code Regular Regular WChar Same as key
English Country Region Name Regular Regular WChar Same as key
French Country Region Name Regular Regular WChar Same as key
Geography Key Key Regular Integer Same as key
Ip Address Locator Regular Regular WChar Same as key
Postal Code Regular Regular WChar Same as key
Sales Territory Key Regular Regular Integer Same as key
Spanish Country Region Name Regular Regular WChar Same as key
State Province Code Regular Regular WChar Same as key
State Province Name Regular Regular WChar Same as key

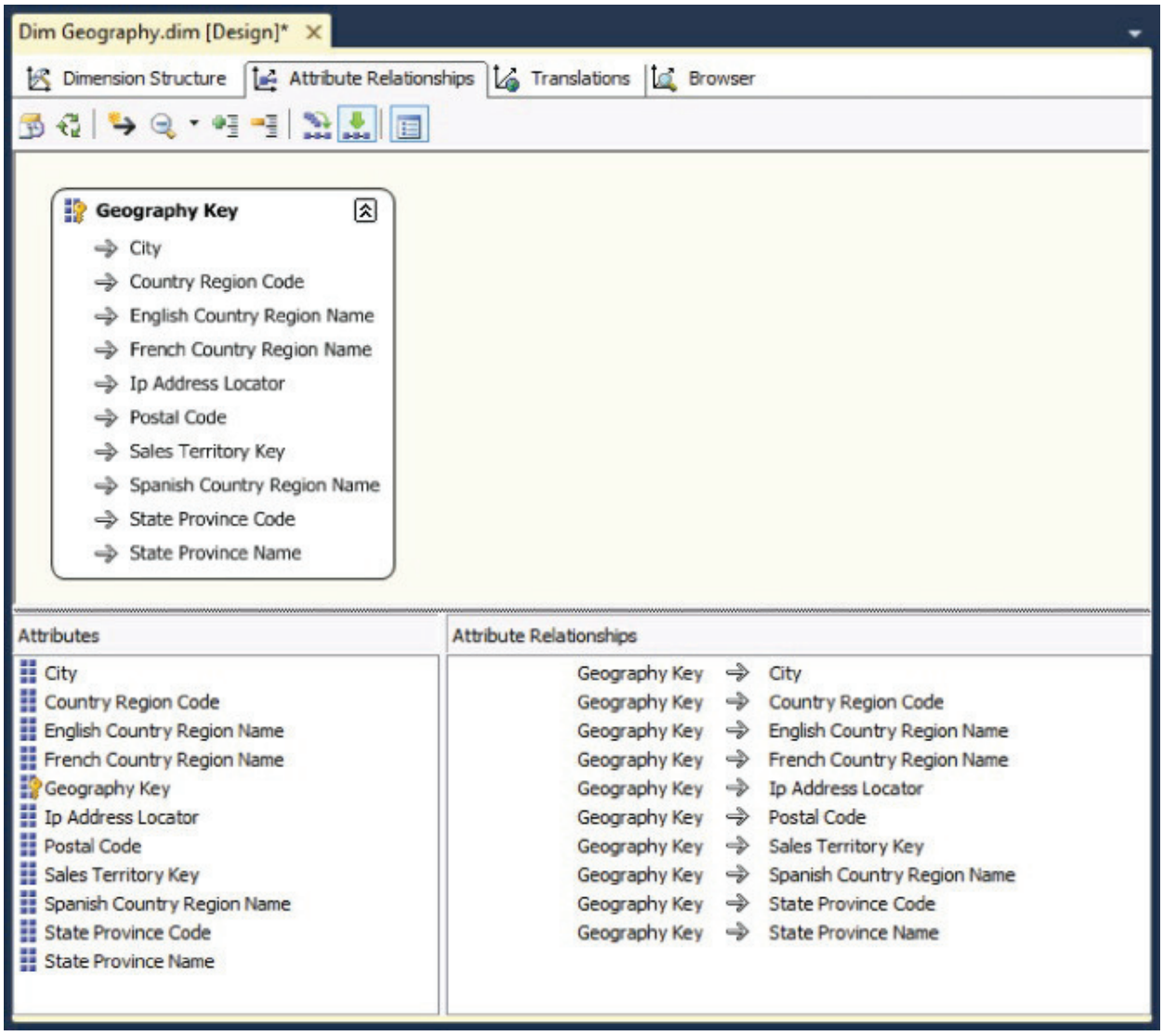
Attribute Relationships

Attribute relationships can be defined when attributes within the same dimension have a one-to-many relationship with each other. For example, if you have the attributes Country, State, and City, you have one-to-many relationships between Country and State, as well as between State and City. Each dimension must have at least one attribute defined as the key attribute. By definition, the key attribute is on the "many" side of a one-to-many relationship with every attribute in the dimension. The Dimension Wizard automatically establishes these relationships with all the attributes of the dimension.

If you are aware of one-to-many relationships between attributes, you should specify those relationships in the Dimension Designer. Specifying attribute relationships helps improve query performance as well as informing the aggregation design for user hierarchies. Because the Dim Geography dimension contains one-to-many relationships, you can specify attribute relationships to get improved query performance. You learn more about the benefits of attribute relationships in Chapter 10.

To view and edit a dimension's attribute relationships you use the Dimension Designer's Attribute Relationships page, as shown in Figure 5.13. The Attribute Relationships page contains three panes. The top pane, called the Diagram pane, graphically shows the defined attribute relationships; the Attributes pane on the lower left shows the list of attributes in the dimension; and the Attribute Relationships pane in the lower right shows the list of defined relationships. The attributes shown in the Diagram pane when you open the designer are all the attributes of the dimension that have a one-to-many relationship with the key attribute (Geography Key). These attributes are also referred to as member property attributes or related attributes. Because an attribute relationship is defined between each member and the Geography Key attribute, you can retrieve the related attributes as properties using the member property MDX functions.

Figure 5.13



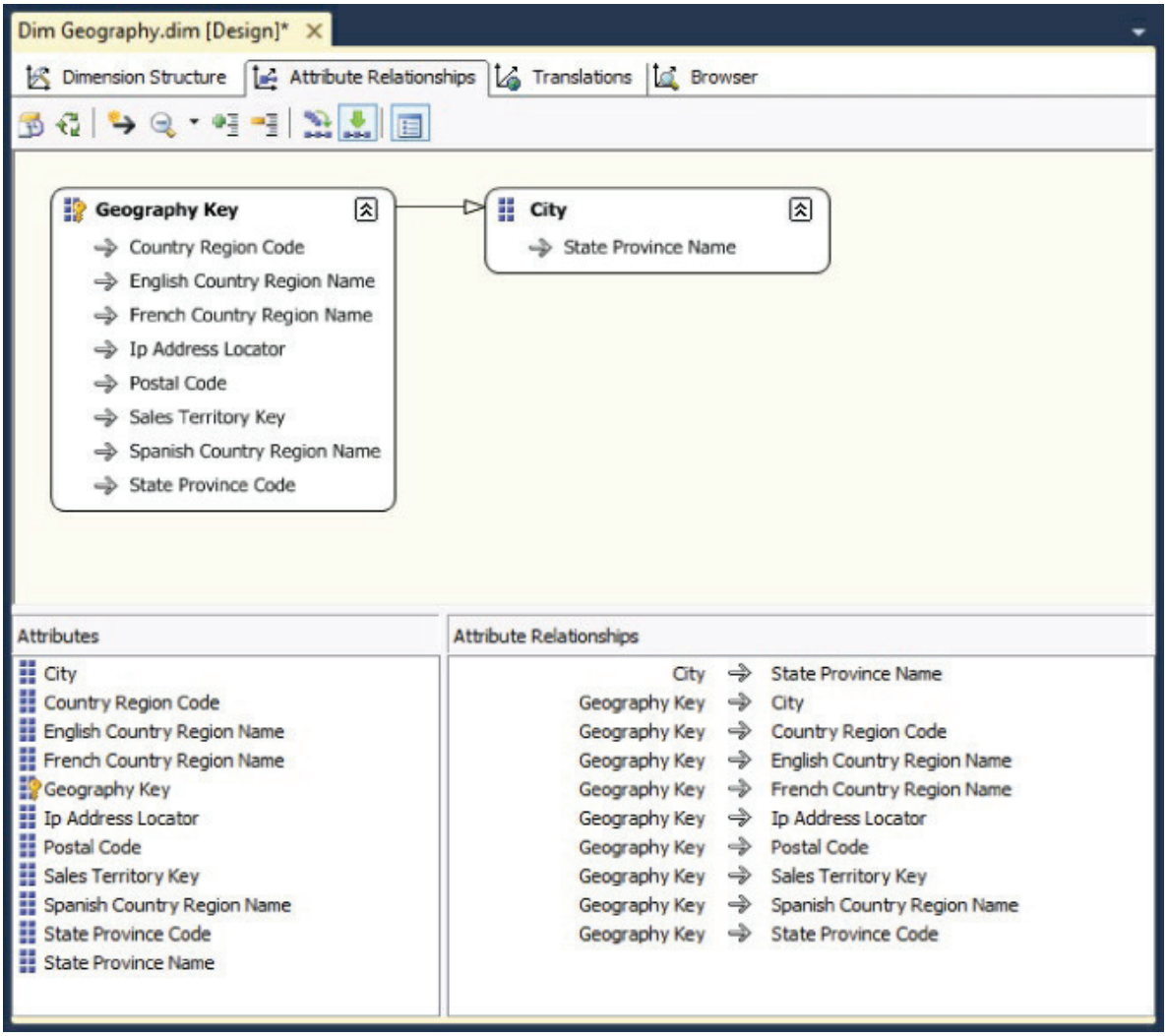
You can create new or modify existing attribute relationships using various methods within each of these panes. Follow these steps to update the Geography dimension's attribute relationships:

1. In the Diagram pane, you can modify attribute relationships by dragging and dropping attributes onto the attribute to which

they are related.

For example, State Province Name has a one-to-many relationship with City. Create the relationship as follows: In the Diagram pane, select the City attribute and drag and drop it onto the State Province Name attribute. This creates a new attribute relationship node, as shown in Figure 5.14. Note the change in the Attribute Relationships pane to reflect the newly defined relationship. In the Diagram pane, you can edit and delete attribute relationships by right-clicking a relationship's line and selecting the wanted action from the context menu.

Figure 5.14

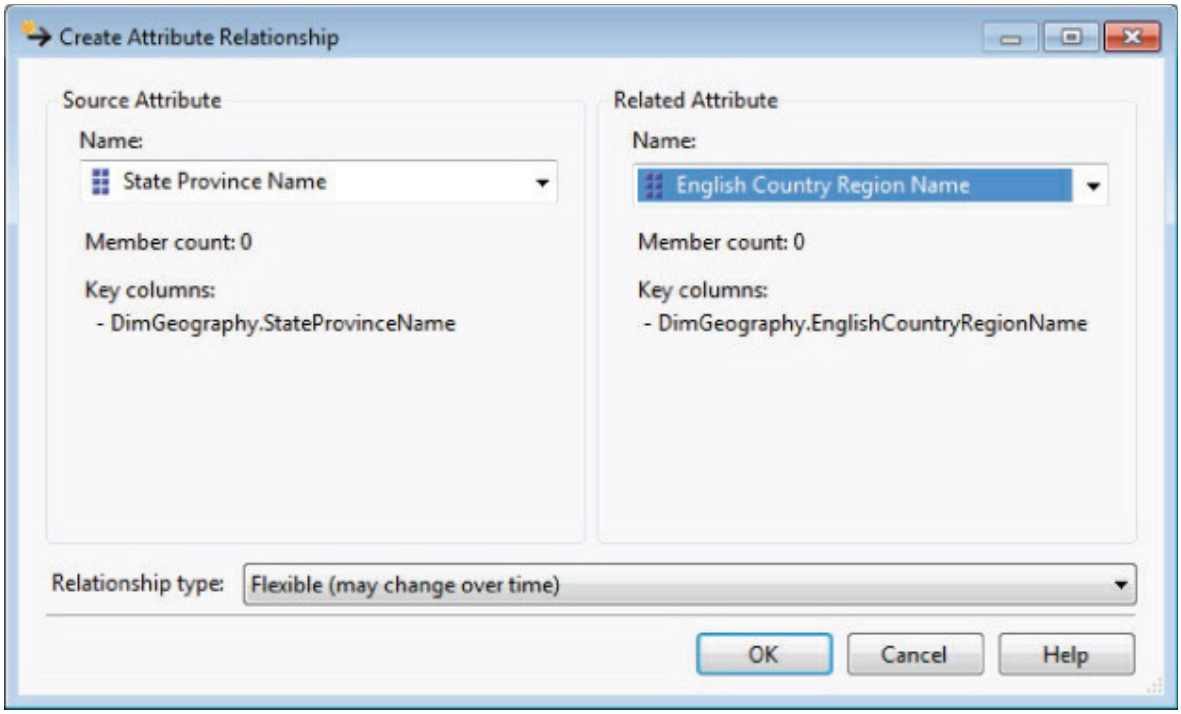


2. To define a new relationship using the Attributes pane, right-click the attribute that makes up the "many" side of the one-to-many relationship, and select New Attribute Relationship. This launches the Create Attribute Relationship dialog. The Source Attribute corresponds to the "many" side of the relationship, and the Related Attribute is the attribute corresponding to the "one" side. You can also set the relationship type in this dialog to either Flexible or Rigid. By default, Analysis Services tools define all relationships to be flexible. A relationship is flexible if the value can change over time. A relationship is rigid if the relationship does not change over time. For example, the birth date of a customer is fixed and hence the relationship between a customer's key/name and the birth date attribute would be defined as rigid. However, the relationship between a customer and his city is flexible because the customer can move from one city to another. In your dimension, there is a one-to-many relationship between the English Country Region Name and State Province Name attributes. To specify that relationship, perform the following steps.

3. In the Attributes pane, right-click the State Province Name attribute, and select New Attribute Relationship from the context menu.

4. In the Create Attribute Relationship dialog, select English Country Region Name as the Related Attribute (see Figure 5.15).

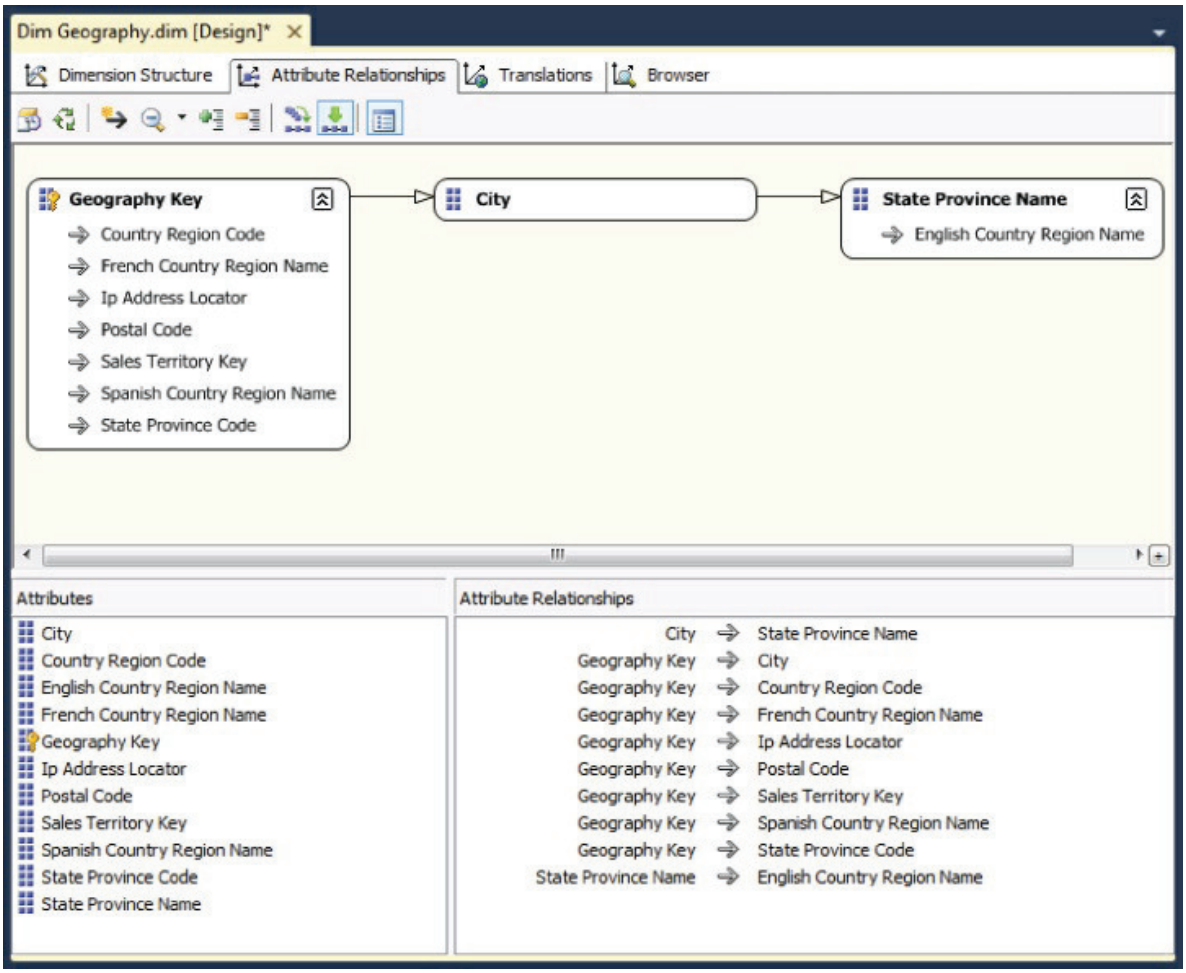
Figure 5.15



5. Click OK to create the relationship.

Your Attribute Relationships display should be similar to that shown in Figure 5.16.

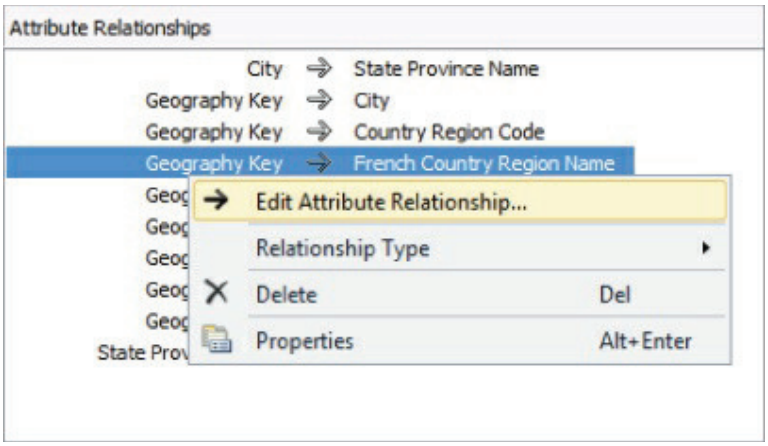
Figure 5.16



To define a new relationship using the Attribute Relationships pane, you must first select an attribute from either the Diagram or

Attributes pane and then right-click in the Attribute Relationships pane in the area not occupied by existing attribute relationships. This can be a bit cumbersome. Use the Diagram pane or the Attributes pane for creating new attribute relationships. Editing and deleting existing relationships in the Attribute Relationships pane, however, is as simple as right-clicking the relationship and choosing the wanted action from the context menu, as shown in Figure 5.17. Editing a relationship launches the Edit Attribute Relationship dialog, whose functionality and layout is identical to the Create Attribute Relationships dialog shown in Figure 5.15; only the title is different.

Figure 5.17



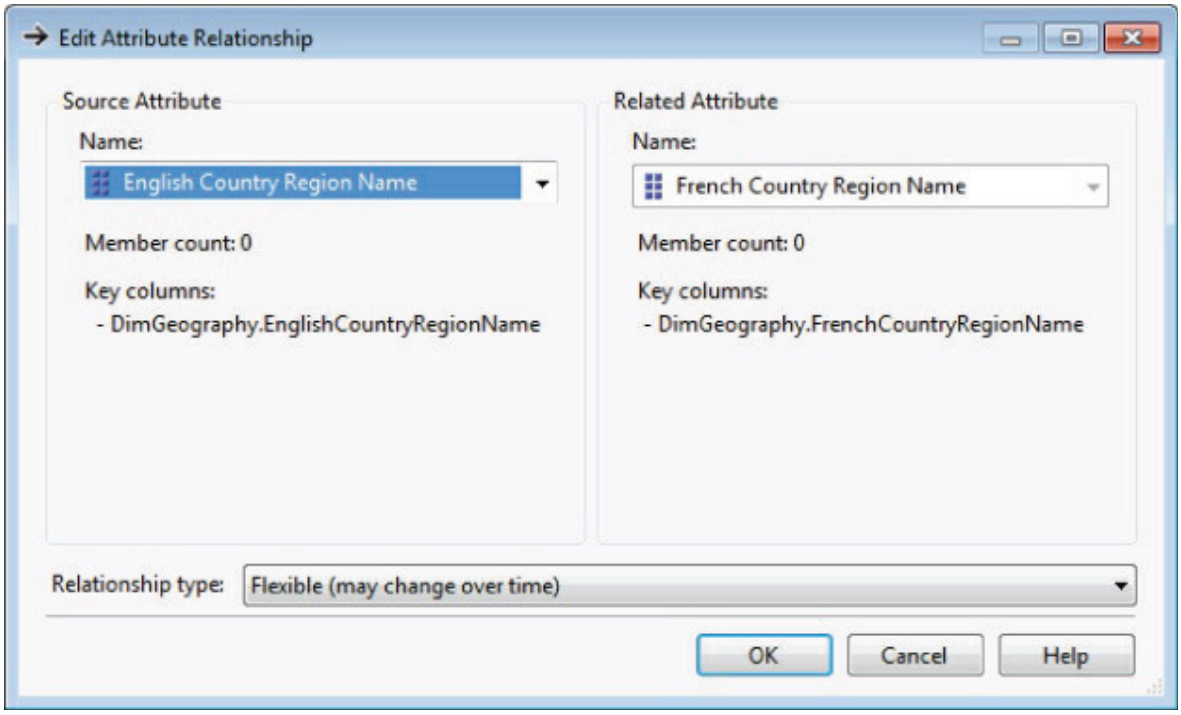
Use the Attribute Relationships pane to edit the existing relationships for the French and Spanish Country Region Name attributes using the following steps:

6. Select the Geography Key to French Country Region Name relationship in the Attribute Relationships pane.

7. Right-click and select Edit Attribute Relationship.

8. In the Edit Attribute Relationship dialog, select the English Country Region Name as the Source Attribute (see Figure 5.18).

Figure 5.18



9. Click OK to save the change to the relationship.

10. In the Properties window, change the Cardinality property corresponding to the relationship between English Country Region Name and French Country Region Name from Many to One.

Repeat the previous five steps for the Spanish Country Region Name attribute relationship.

You have now used three different methods for working with attribute relationships. Often in business analysis when you analyze a specific member of a dimension, you need to see the properties of the dimension member to understand it better. In such circumstances, instead of traversing the complete hierarchy, you can retrieve the member by querying the member properties. This, once again, is a

performance improvement from the end user's perspective. A wide variety of client tools support the ability to retrieve member properties of a specific member when needed by the data analyst. You can add additional attributes by dragging and dropping a column from the DSV to the Attributes pane or delete an existing attribute by right-clicking that attribute and selecting Delete.

User Hierarchies

User hierarchies (also called multilevel hierarchies) are created from the attributes of a dimension. Each user hierarchy contains one or more levels, and each level is an attribute hierarchy itself. Based on the attributes of the Geography dimension you created, a logical user hierarchy to create would be Country-State-City-Postal Code. You can create this hierarchy using the following steps:

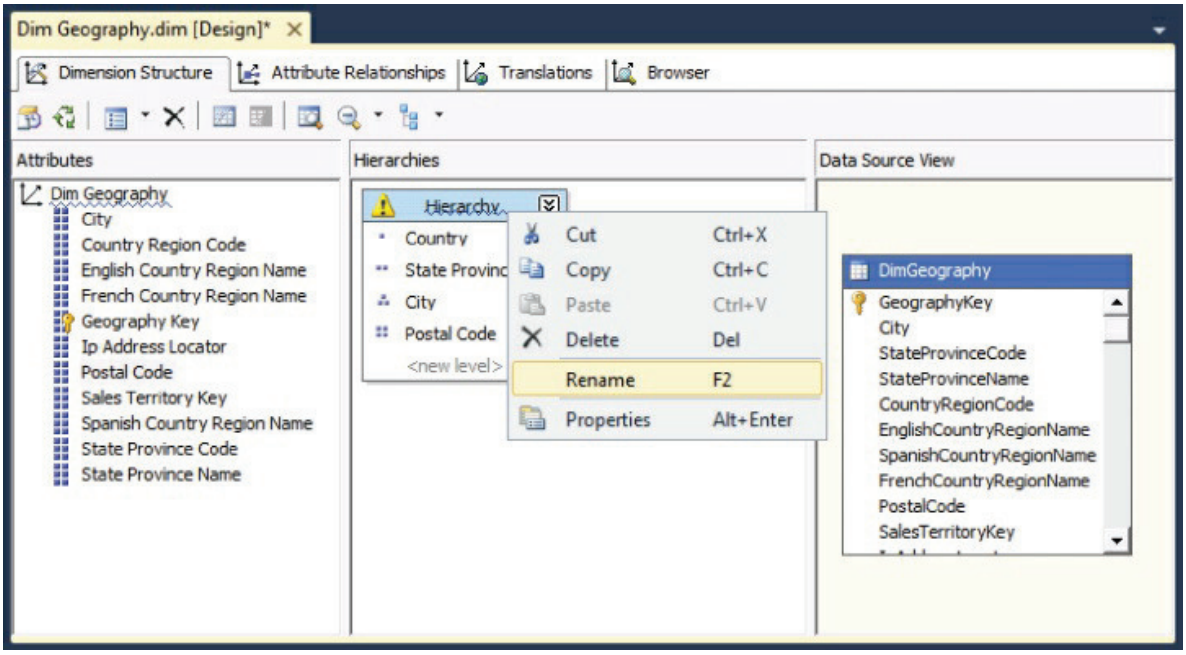
1. Switch to the Dimension Structure tab of the Dim Geography dimension, and drag and drop the English Country Region Name attribute from the Attributes pane to the Hierarchies pane. This creates a user hierarchy called Hierarchy with one level: English Country Region Name. This level corresponds to a country. To make this name more user-friendly, rename the English Country Region Name attribute to **Country** by right-clicking the attribute within the user hierarchy and selecting Rename.

2. Drag and drop State Province Name from the Attributes pane to the Hierarchies pane such that the State Province Name attribute is below Country in the user hierarchy. Rename State Province Name to **State Province** by right-clicking the attribute and selecting Rename.

3. Drag and drop City and Postal Code attributes to the user hierarchy in that order so that you now have a four-level hierarchy Country-State ProvinceCity-Postal Code.

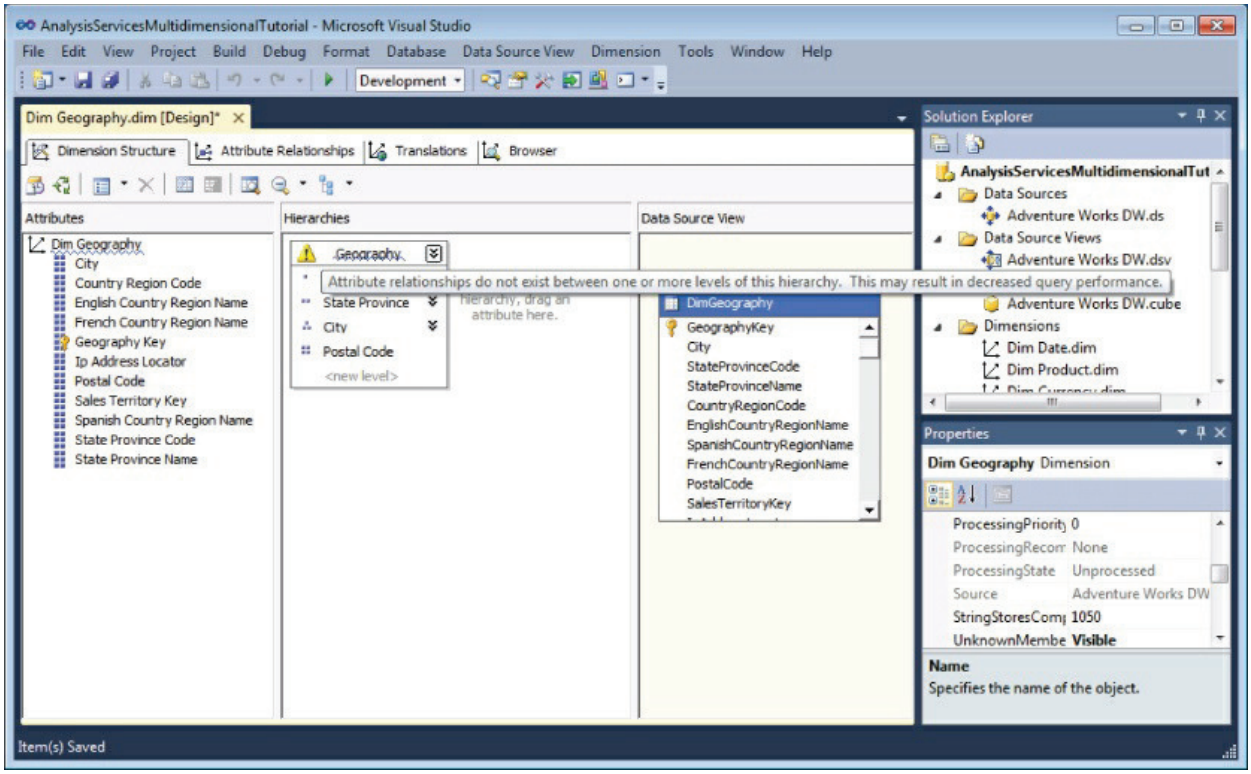
4. The default name of the hierarchy you have created is Hierarchy. Rename it to **Geography** by right-clicking its name and selecting Rename (see Figure 5.19). You can also rename hierarchy and level names in the Hierarchies pane by selecting the item and changing the value of its Name property in the Properties pane.

Figure 5.19



You have now created a user hierarchy called Geography that has four levels, as shown in Figure 5.20. You can click the arrows to expand the attribute in each level to see its related attributes.

Figure 5.20



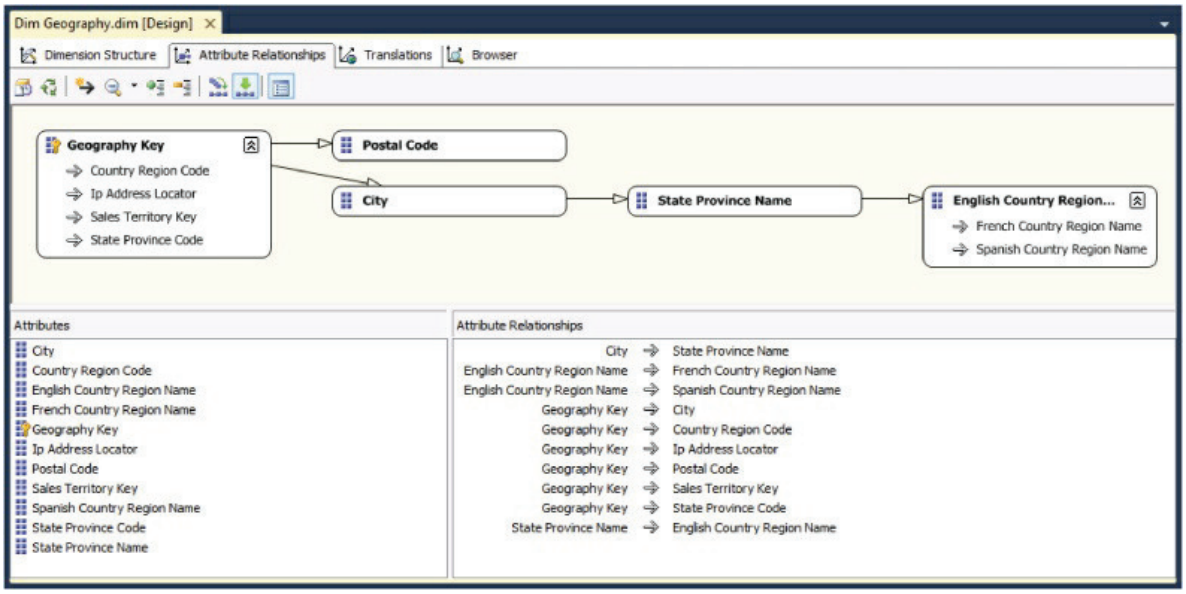
Notice the Warning icon next to the Geography hierarchy name and the squiggly line under the name of the hierarchy. If you place your mouse over this icon or the hierarchy name, you can see a tooltip message indicating that attribute relationships do not exist between one or more levels of the hierarchy and could result in decreased performance.

The current hierarchy design is called an unnatural hierarchy. An unnatural hierarchy exists when knowing the attribute value of one level of the hierarchy is not sufficient to know who its parent is in the next level up the hierarchy. Another example of an unnatural hierarchy would be a Customer Gender-Age hierarchy, where Gender is the top level of the dimension and Age is the second level. Knowing that a customer is 37 years old does not give any indication of gender.

Conversely, in a natural hierarchy, knowing the value of an attribute at one level clearly indicates who its parent is on the next level of the hierarchy. An example of a natural hierarchy would be a Product dimension hierarchy with Category, Sub-Category, and Product levels. By knowing that a product is in the Mountain Bike Sub-Category, you would know that it belongs to the Bike Category. This relationship between attribute values is defined through attribute relationships. For a hierarchy to be considered natural, attribute relationships must exist from the bottom level of the hierarchy all the way to the top. Analysis Services materializes only hierarchies considered natural. Use the following steps to refine the current Geography hierarchy so that it is natural:

1. Switch back to the Attribute Relationships page. The page should look similar to Figure 5.21.

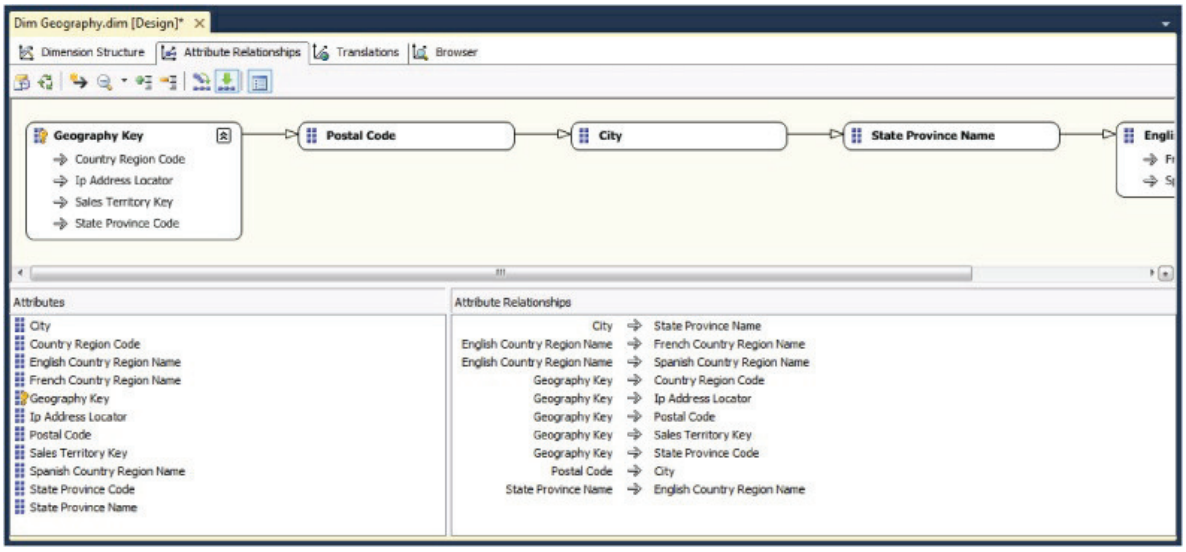
Figure 5.21



2. There is no relationship between Postal Code and City. In the Diagram pane, drag and drop the Postal Code attribute to the City attribute.

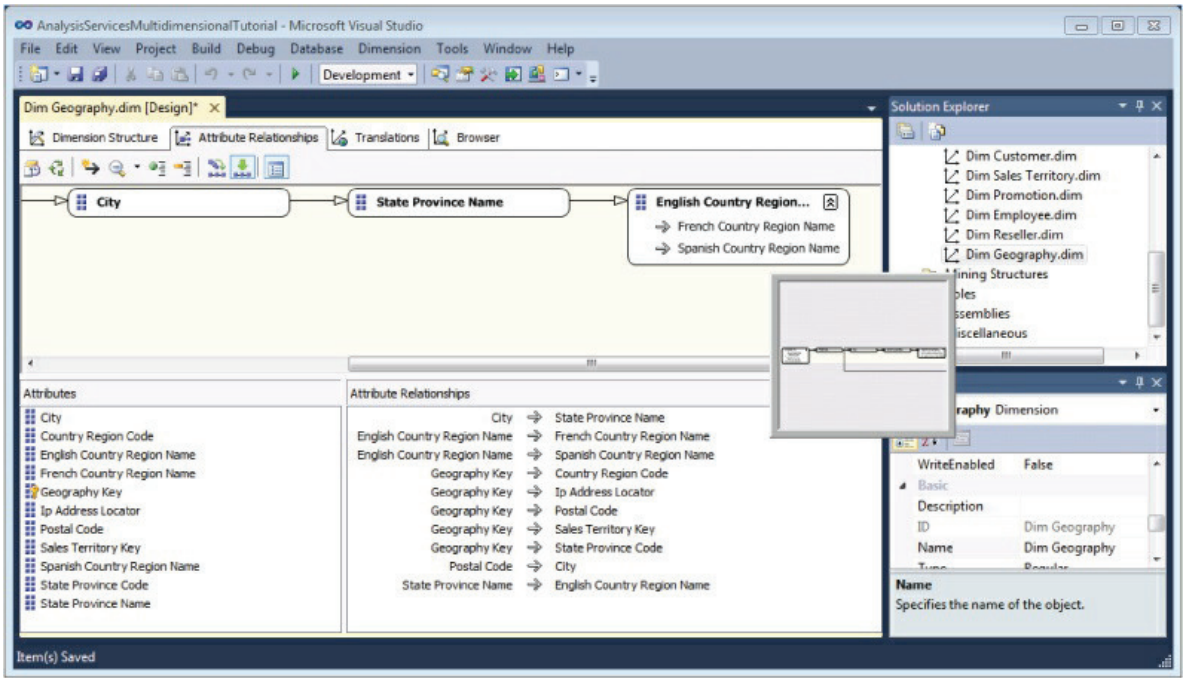
An attribute relationship between the Postal Code attribute and the City attribute is created, as shown in Figure 5.22.

Figure 5.22



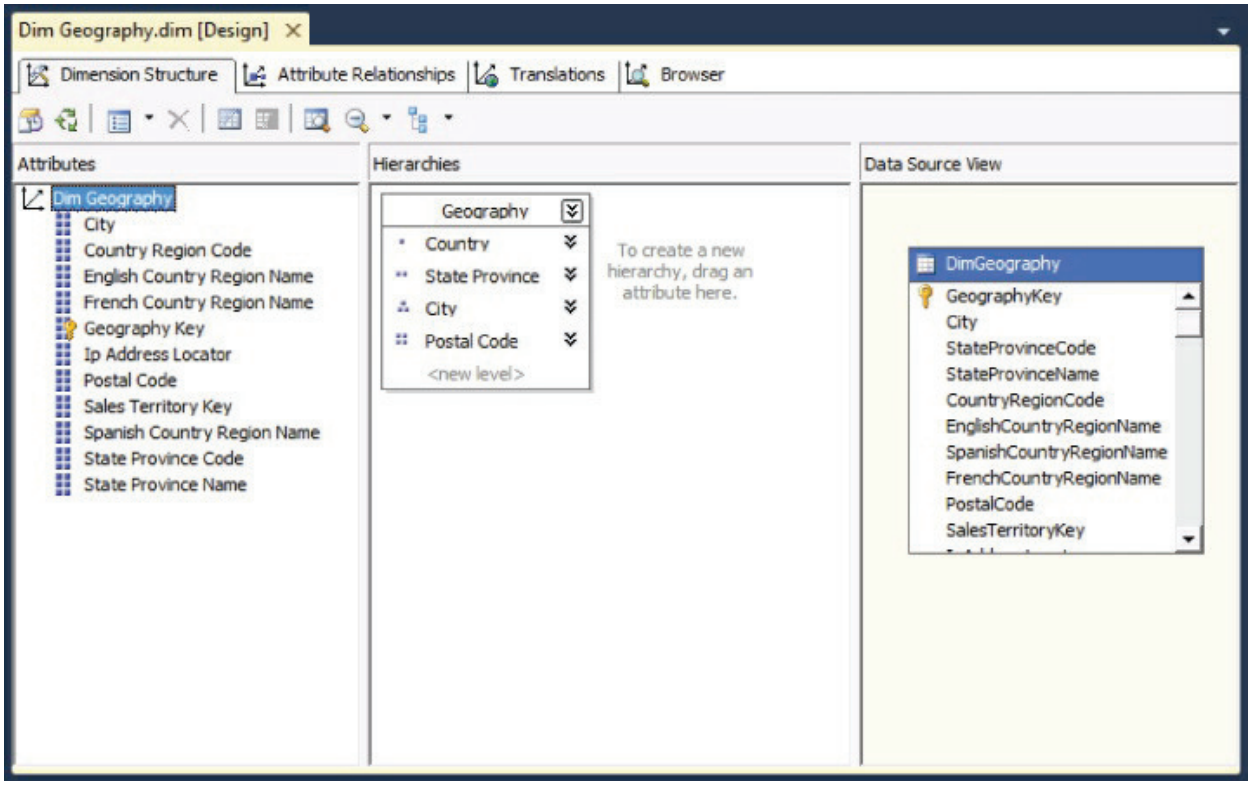
The visualization of the attribute relationships extends beyond the Diagram pane. (Depending on the resolution of your monitor, you might view all the attribute relationships.) You can zoom in or zoom out using the Zoom item in the context menu of the visualization pane to adjust the size of the content of the Diagram pane to see all the attribute relationships. Sometimes the attribute relationships view can be quite large, depending on the number of attributes and the relationships you have established. You can easily navigate to the area of the visualization pane you're interested in by clicking the "+" symbol at the far right of the horizontal scrollbar and using the locator window (as shown in Figure 5.23).

Figure 5.23



3. Switch back to the Dimension Structure tab, verify the warning is gone, and save the dimension, as shown in Figure 5.24.

Figure 5.24



Browsing the Dimension

After successfully defining the Dim Geography dimension, you definitely would like to see the results of what you have created and find out how you can view the members of the dimension. At this point the dimension has been designed but not deployed to the server. Indeed, there has been no interaction with the instance of Analysis Services yet. To see the members of the dimension, Analysis Services needs to receive the details of the dimension (the attributes, member properties, and the multilevel hierarchies you have created).

The Analysis Services tools communicate to the instance of Analysis Services via XMLA (XML for Analysis). XMLA is an industry-standard, Simple Object Access Protocol (SOAP)–based XML Application Programming Interface (API) designed for OLAP and Data Mining. The XMLA specification defines two functions, Execute and Discover, which send actions to and retrieve data from the host instance. The Execute and Discover functions take several parameters that define the actions that the instance of Analysis Services can perform. One of the parameters of the Execute function is the command sent to an instance of Analysis Services. In addition to supporting the functionality defined in the XMLA specification, Analysis Services supports extensions to the standard. Following is a sample Execute request sent to an instance of Analysis Services using XMLA.

```
<Execute xmlns="urn:schemas-microsoft-com:xml-analysis">
  <Command>
    <Statement>
      SELECT Measures.MEMBERS ON COLUMNS FROM [Adventure Works DW]
    </Statement>
  </Command>
  <Properties>
    <PropertyList>
       <Catalog>AnalysisServicesMultidimensionalTutorial</Catalog>
       <Format>Multidimensional</Format>
       <AxisFormat>ClusterFormat</AxisFormat>
   </PropertyList>
 </Properties>
</Execute>
```

In the preceding XMLA, a request is sent to execute an MDX query specified within the `Statement` command on the AnalysisServicesMultidimensionalTutorial catalog. The request shown results in the query being executed on the server side and the results returned to the client via XMLA.

Several different commands communicate to the Analysis Services Server. Some of the common ones are Create, Alter, Process, and Statement. These commands change the structure of objects referenced in the command. Each Analysis Services object has a well-defined set of properties. The definition of the objects is accomplished by commands referred to as Data Definition Language (DDL) commands.

Other commands work with data that has already been defined. Those commands are referred to as Data Manipulation Language (DML) commands. You learn some of the DML and DDL commands used in Analysis Services in various chapters of the book through examples. For an in-depth understanding of DML and DDL commands, see the Analysis Services documentation.

You might recall that you deployed the AnalysisServicesMultidimensionalTutorial project in Chapter 2. When you deploy a project, SSDT packages the design change information in the project into XMLA requests and sends them to the server. At this point, you might want to see the contents of the dimension you have created. To do this you need to deploy the project to an instance of Analysis Services. When you deploy the project to the Analysis Services server, several XMLA requests are sent:

1. Request a list of the databases from Analysis Services to determine if the database defined by the current project already exists on the instance. The project name you specified will be used as the database name. SSDT sends either a Create or Alter command for the database based on whether the database already exists on the server. Based on the deployment settings in your project, SSDT then sends either the entire definition of all the objects in your project or only the changes you have made since the last deploy. We have not included the contents of the Create/Alter XMLA request in the following code because it is quite large. You can use the SQL Server Profiler to view the XMLA requests sent by SSDT. (You learn to use SQL Server Profiler in Chapter 11.)

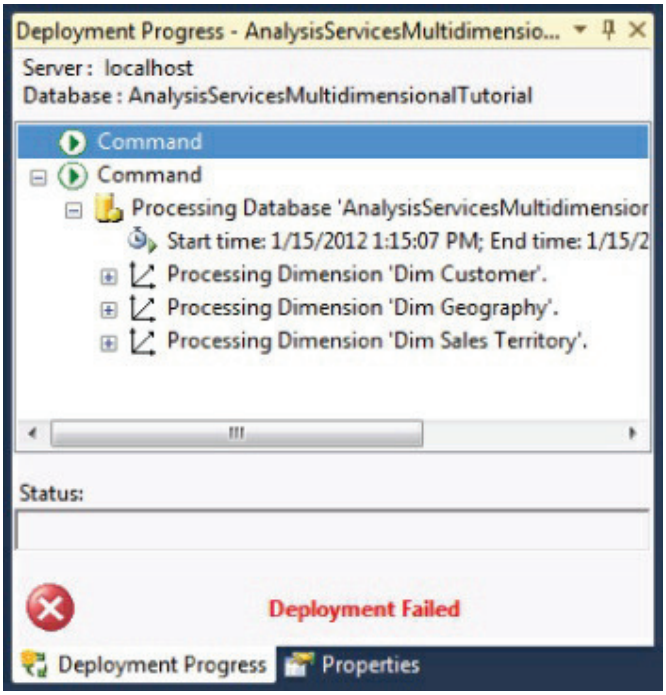
2. SSDT then sends an XMLA request to process the objects you have created and/or modified. Following is the request sent to the server to process the Dim Geography dimension:

```
<Batch xmlns="http://schemas.microsoft.com/analysisservices/2003/engine">
   <Process>
     <Type>ProcessDefault</Type>
     <Object>
       <DatabaseID>AnalysisServicesMultidimensionalTutorial</DatabaseID>
     </Object>
   </Process>
 </Batch>
```

SSDT performs validations to make sure your dimension design is correct. If there are errors, SSDT shows those errors using red squiggly lines. You saw an example of this earlier in the chapter when you created a user hierarchy and hadn't yet defined attribute relationships between the hierarchy's levels. In addition to that, a set of error handling properties in the Analysis Services instance helps in validating your dimension design at deployment and processing time. SSDT sends the error handling property information to the Analysis Services instance to indicate how the server should respond to any referential integrity errors as part of the deployment process.

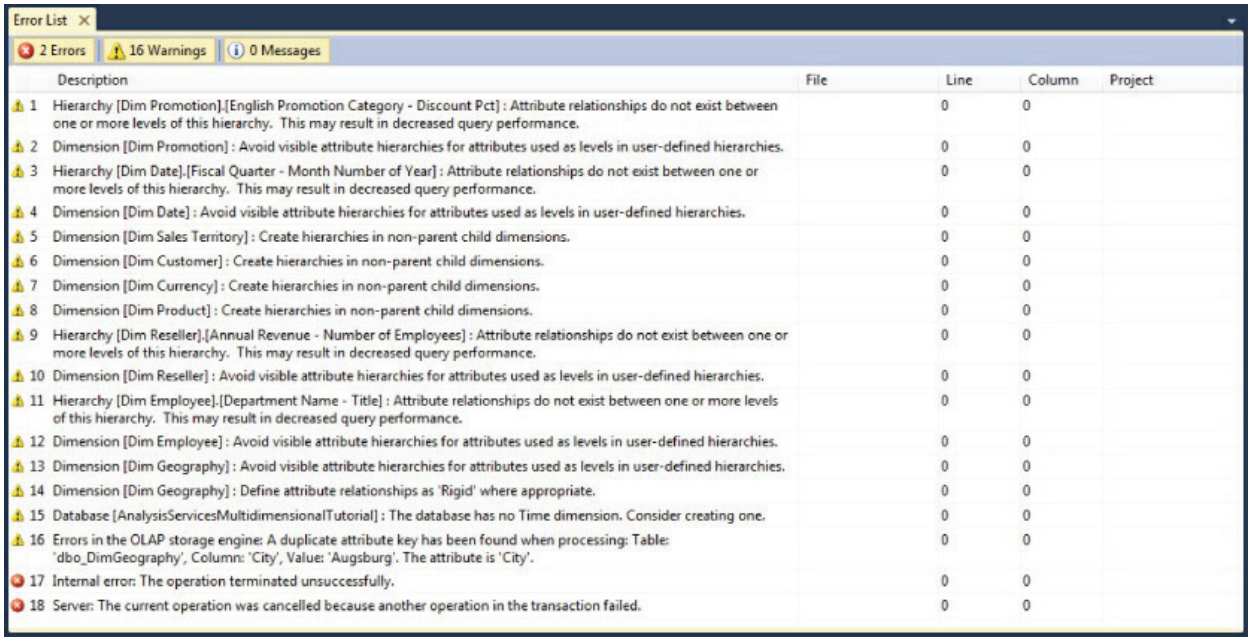
Deploy the AnalysisServicesMultidimensionalTutorial project to your Analysis Services instance by either pressing the F5 key or right-clicking the project in the Solution Explorer window and selecting Deploy. SSDT attempts to deploy the project to the Analysis Services instance. Deployment will fail, as you can see in the Deployment Progress window shown in Figure 5.25.

Figure 5.25



SSDT reports all the warnings and errors identified by SSDT from the Analysis Services instance using the Error List window, as shown in Figure 5.26.

Figure 5.26



The first 15 warnings shown in Figure 5.26 are an example of the Best Practices warnings feature mentioned earlier. This feature highlights common violations of best practices that the Analysis Services team has historically seen and are, by necessity, general. Some of these warnings shown by SSDT might be the result of valid design decisions you made for your particular Analysis Services database. Hence SSDT supports a warning infrastructure by which you can disable warnings for specific objects or even disable specific warnings from reappearing in future deployments. When you right-click one of the first 15 warnings, you can see an option to dismiss the warning. You cannot dismiss a warning reported by the Analysis Services instance (as opposed to SSDT) or any error. If you click the sixteenth warning, you can see that the warning cannot be dismissed. This is the first warning reported by the Analysis Services instance followed by the errors that fail the deployment of your AnalysisServicesMultidimensionalTutorial project. You learn more about the Best Practices warning feature in Chapter 9.

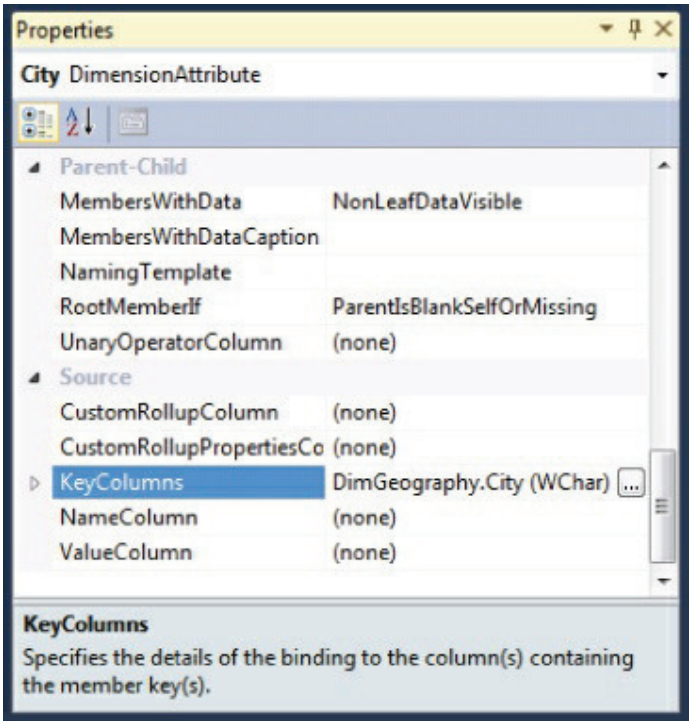
The Analysis Services instance warning (warning 16, shown in Figure 5.26) indicates that, while processing the City attribute, duplicate attribute key values were identified. This warning indicates that there are multiple cities with the same name. This error is raised because you have defined and guaranteed a one-to-many attribute relationship between the City attribute and the State Province Name attribute. The Analysis Services instance identifies that there are several different cities with the same name and cannot decide which State Province Name has the relationship to a specific city. If you query the City column in the DimGeography table, you can see that City names are not unique. For example, London, Paris, and Berlin all appear in excess of a dozen times in the DimGeography table of the AdventureWorksDW database. Hence the Analysis Services instance raises the error with the text "Errors in OLAP Storage Engine." Due to this error, the Analysis Services instance fails the processing of the City attribute, and subsequently the Dim Geography dimension, and the deployment fails.

To correct this issue, you need to make sure each city is unique. One way you can do this is by creating a surrogate key for the city that makes each city unique and use that key as the key column for the City attribute. Alternatively, you can use composite keys to uniquely identify a City attribute. In the following steps you use the second approach and build composite keys from multiple attributes that uniquely identify all members of those attributes that are not unique by themselves. To uniquely identify a city, you need to know the State Province Name to which it belongs. Therefore, the composite key for the City attribute should include the City and State Province Code. Follow these steps to make the City attribute have unique members:

1. Open the Dim Geography dimension in the Dimension Designer, and select the City attribute in the Attributes pane.

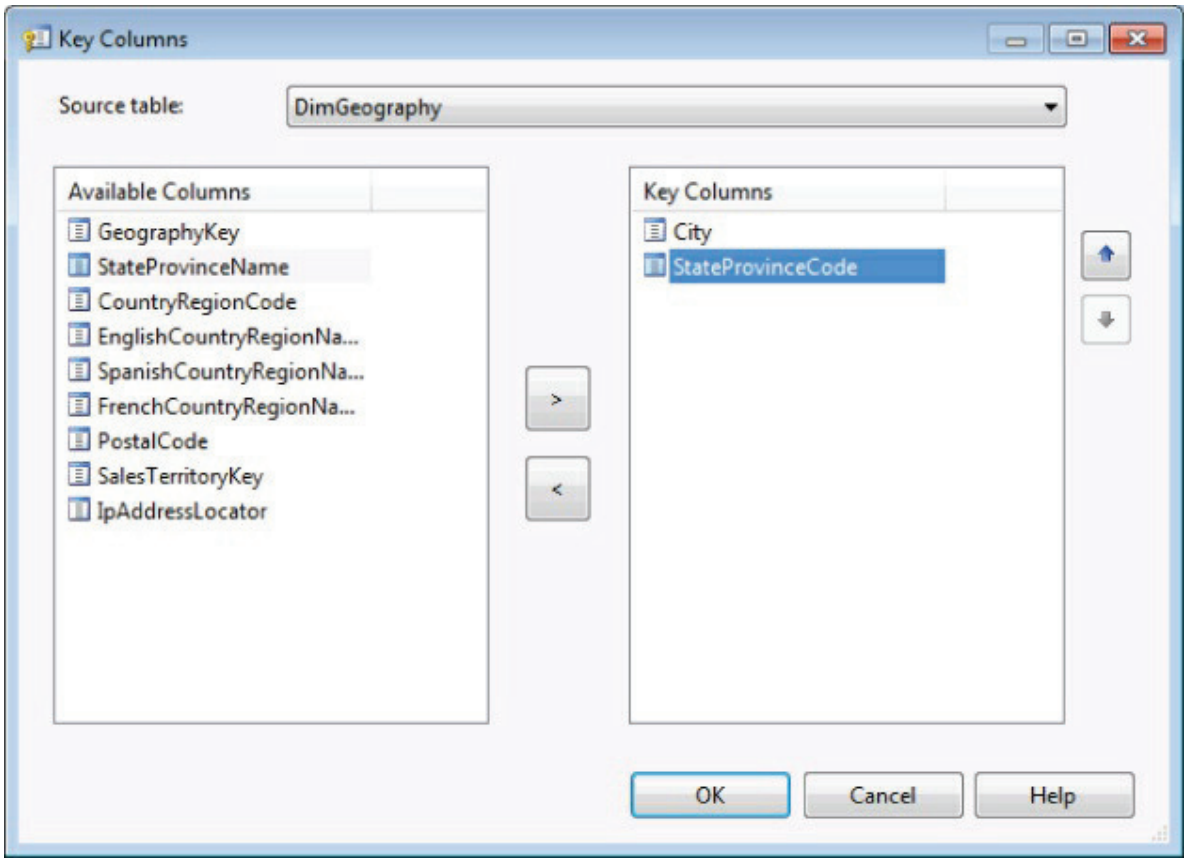
2. In the Properties pane, locate the KeyColumns property, and click the ellipses (as shown in Figure 5.27) to open the Key Columns selection dialog.

Figure 5.27



3. In the Key Columns selection dialog, add the StateProvinceCode to the list of Key Columns, as shown in Figure 5.28. Click OK.

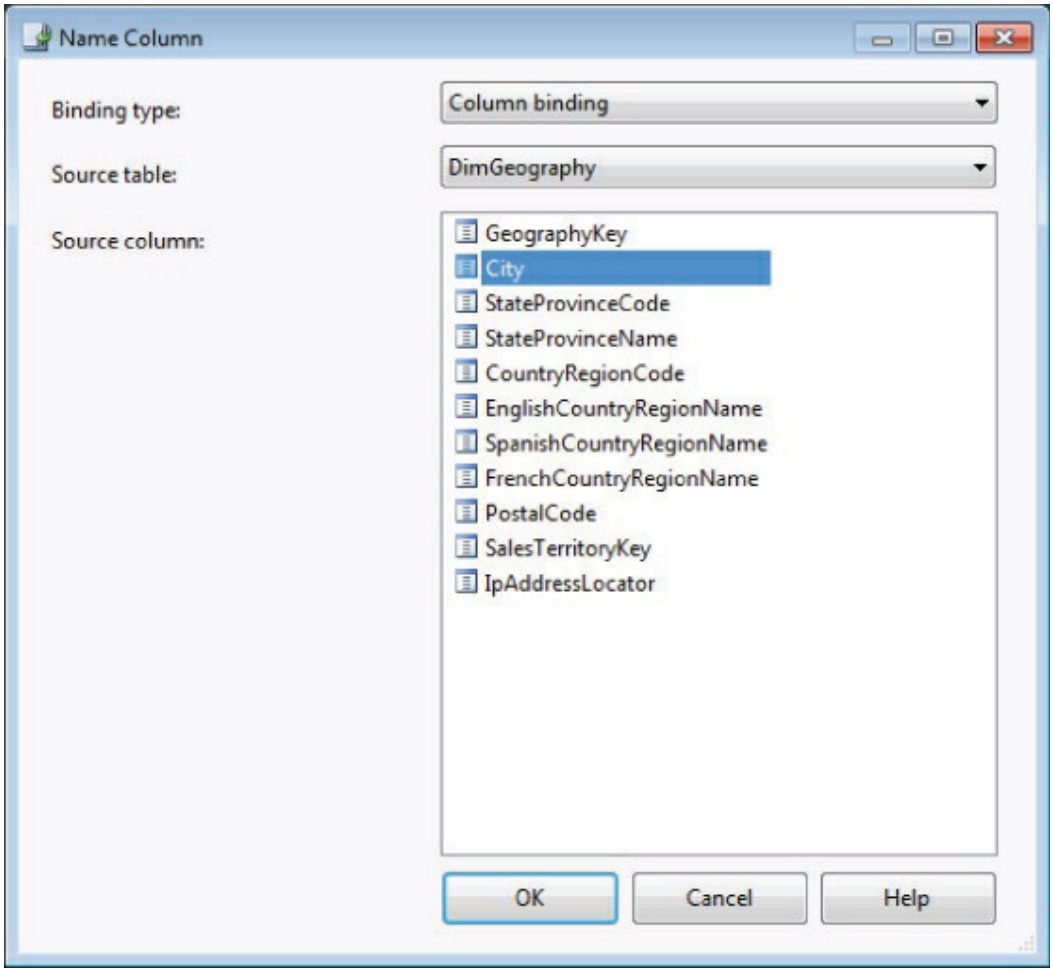
Figure 5.28



By default, the Dimension Wizard uses the column name as the key column for the attribute. The Analysis Services instance automatically infers the same column to be the name column (the column used to display the member names for the attribute). Whenever you define a composite key, you need to define a name column for the attribute because SSDT and the Analysis Services instance do not know which of the composite key columns should be used as the name column for the attribute.

4. In the NameColumn property for the City attribute, click the ellipses to open the Name Column selection dialog (shown in Figure 5.29) and select City as the source for the name of the City attribute. Click OK.

Figure 5.29



The DimGeography table in the data source also contains duplicate postal codes. As you just did for the City attribute, you need to make the Postal Code attribute members unique.

5. Select the Postal Code attribute in the Dimension Designer's Attributes pane.

6. In the Properties pane, locate the KeyColumns property, and click the ellipses to open the Key Columns selection dialog.

7. Change the key columns for the Postal Code attribute to include the StateProvinceCode, City, and PostalCode columns from the data source. Click OK.

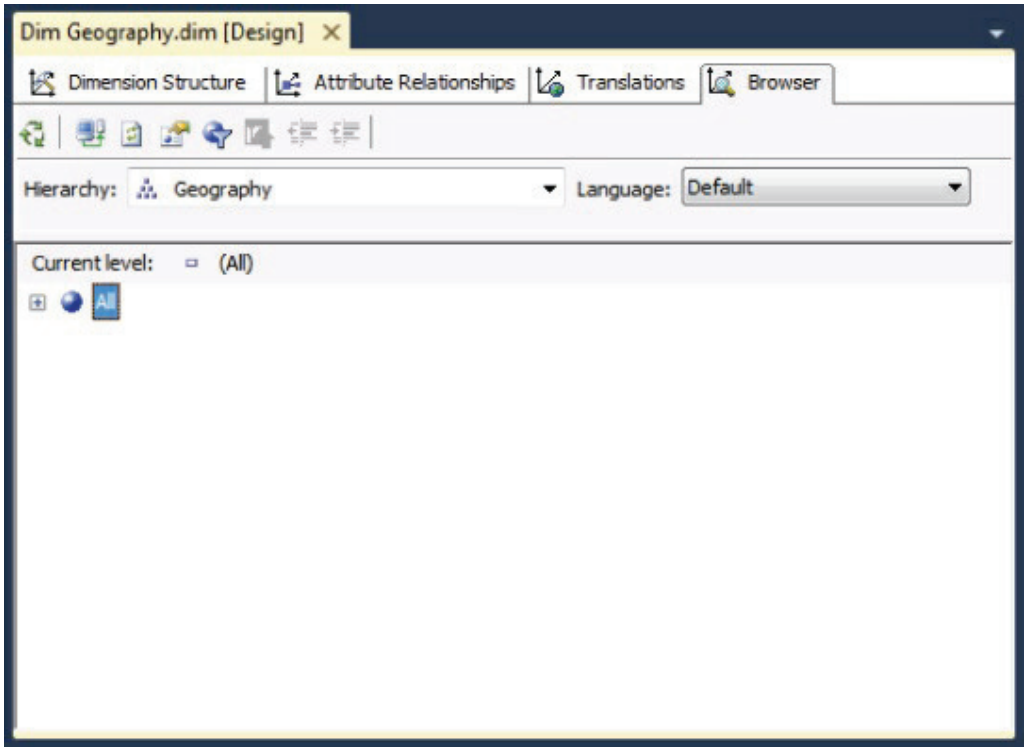
8. Change the NameColumn property by clicking the ellipses next to the NameColumn property in the Properties window.

9. In the Name Column dialog, set the NameColumn property to PostalCode. Click OK.

10. Deploy the AnalysisServicesMultidimensionalTutorial database to the Analysis Services instance.

The AnalysisServicesMultidimensionalTutorial database now successfully deploys. Now that you have successfully deployed the database, you can browse the data for the Dim Geography dimension by switching to the Browser tab of the Dimension Designer, as shown in Figure 5.30. To display the data in the browser, SSDT sends several Discover requests to retrieve information, such as the hierarchies and levels available for the dimension. Finally, an MDX query is sent to the server by SSDT to retrieve dimension data. The MDX query follows:

Figure 5.30




```
Select Head( [Dim Geography].[Geography].Levels(0).Members, 1000 ) on 0
from [$Dim Geography]
```

Because you have some familiarity with MDX, you might have deciphered most of the query. This query uses the `HEAD` function to request the first 1,000 members from Level 0 of the Geography hierarchy in the Dim Geography dimension. In the `FROM` clause you see [$Dim Geography]. In general, an MDX `FROM` clause should contain a cube name, but in this query the `from` clause specifies a dimension name, so how does this MDX query work? Here's how: When a dimension is created, the server internally stores the values of the dimension as a cube. This means that *every dimension is internally represented as a cube* with a single dimension that holds all the attribute values. The dimension you have created as part of the AnalysisServicesMultidimensionalTutorial database is called a database dimension. Because each database dimension is a one-dimensional cube, they can be queried with MDX using the special character $ before the dimension name. This is exactly what you see in the query: [$Dim Geography].

The hierarchy first shown in the dimension browser is the most recently created user hierarchy (in this case, Geography). You can choose to browse any of the user hierarchies or attribute hierarchies in a dimension by selecting them from the drop-down Hierarchy list. This list contains the dimension's user hierarchies followed by its attribute hierarchies. Each attribute hierarchy and user hierarchy within a dimension has a level called the All level. In Figure 5.30 you can see the All level for the Geography hierarchy. The All level is the topmost level of most hierarchies. (It can be removed in certain cases.) You can change the name of the All level by changing the AllMemberName property of the hierarchy. It makes sense to call the level "All" because it encompasses all the sublevels in the hierarchy. If a hierarchy does not contain the All level, the members of the topmost level display as the first level in the Dimension Designer's Browser page.

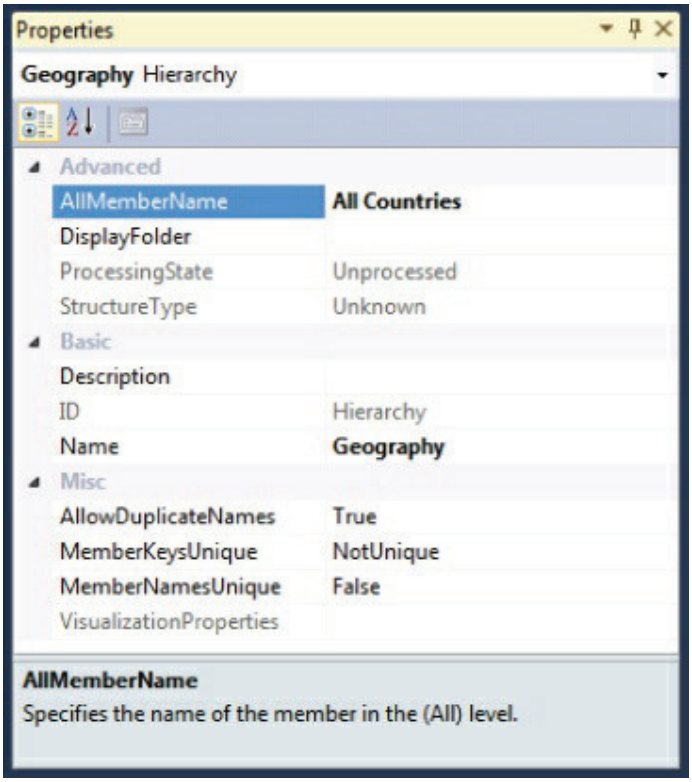
Assume you want to change the name of the All level of the Geography hierarchy to All Countries. The following steps show how to do this:

1. Go to the Dimension Structure page of the Dimension Designer.

2. Select the Geography hierarchy in the Hierarchies pane.

3. The Properties window now shows all the properties of this hierarchy. The first property is AllMemberName and it displays no value. Add a value by entering All Countries in the text box to the right of AllMemberName, as shown in Figure 5.31.

Figure 5.31

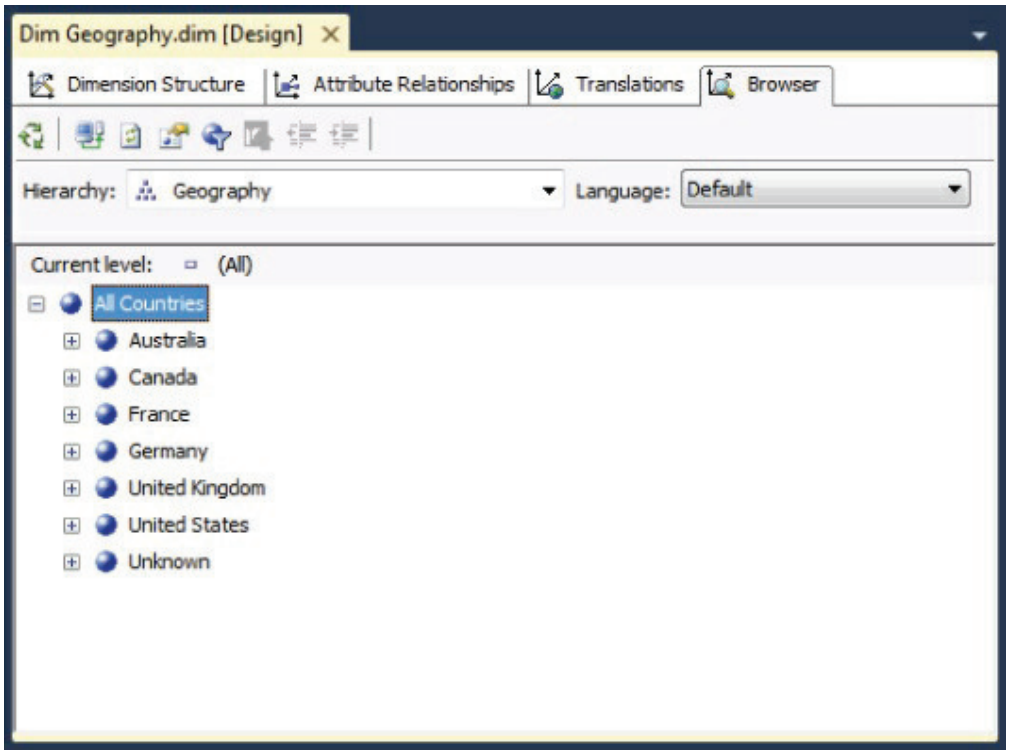


4. Deploy the project.

5. After successful deployment, switch from the Dimension Structure page to the Browser page. You see a message in the Dimension Browser to click Reconnect to see the latest changes. Click the Reconnect link.

You can now see that the All level of the Geography hierarchy has changed to All Countries, as shown in Figure 5.32. The figure also shows the All Countries level expanded to show the members in the next level.

Figure 5.32



When you expand the All Countries level, the following MDX query is sent to the Analysis Services instance to retrieve the members in the next level:

```
WITH MEMBER [Measures].[-DimBrowseLevelKey 0-] AS
```

```
'[Dim Geography].[Geography].currentmember.properties("key0", TYPED)'

SELECT { [Measures].[-DimBrowseLevelKey 0-] } ON 0,
Head( [Dim Geography].[Geography].[All Countries].Children, 1000) ON 1
FROM [$Dim Geography]
CELL PROPERTIES VALUE
```

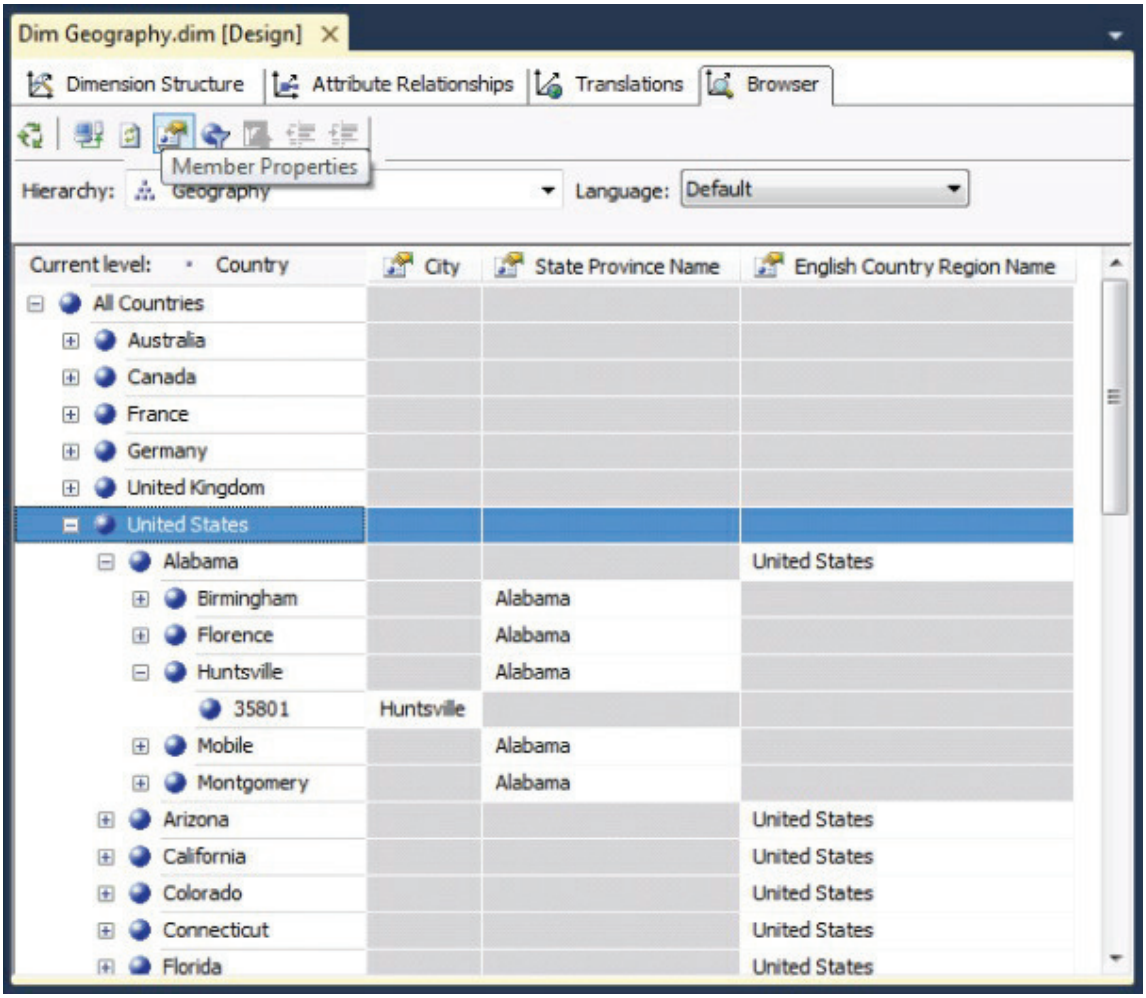
The goal of the MDX query is to retrieve all the members that are children of the All Countries level. Similar to the MDX query that was sent to retrieve members in level 0, this query uses the `HEAD` function to restrict the amount of returned results to a maximum of the first 1,000 children. This query includes a calculated measure called `Measures.[-DimBrowseLevelKey 0-]`, which is selected in the MDX query. The calculated measure expression in this query retrieves the key of the current member by using the MDX `Properties` function. This function returns the value of the member property specified as the first argument to the expression. In this query the value requested is the key of the current member. The property requested is "key0" as opposed to "key." If the key of the member is a composite key, only the first column (column 0) of the composite key will be retrieved. If the query had specified "key" instead of "key0" and the key were a composite key, the value returned would be null.

Other parameters that can be passed to the `Properties` function are `NAME`, `ID`, and `CAPTION`, or the name of a member property or related attribute. The properties `NAME`, `ID`, `KEY`, and `CAPTION` are called *intrinsic member properties* because all attributes and hierarchies will have them. The second argument passed to the `Properties` function is optional, and the only value that can be passed is `TYPED`. If the `Properties` function is called without the second parameter, the function returns the string representation of the property. If the second argument, `TYPED`, is passed to the function, it returns the strongly typed value of the requested property (using the data type defined in the data source). For example, if the first argument is `KEY` and if the key of this attribute is of type integer, the `Properties` function returns the key as an integer value if you specify the `TYPED` parameter. Typically, specifying the second parameter is useful if you want to use the returned property value to filter the results based on a member property. For example, if the key of the Geography hierarchy is an integer and if you want to see only the children of member United States, you can use the `Filter` function along with the strongly typed calculated measure that has been created specifying the `TYPED` parameter.

The Dimension Browser sends the preceding MDX query to the Analysis Services server and uses the returned information to populate the second level of the hierarchy, as shown in Figure 5.32.

When you defined an attribute relationship between the State Province Name and City attributes earlier in the chapter, you implicitly set member properties for those attributes. You can see these member properties in the Dimension Designer's Browser page. To do that you can either click the Member Properties button in the Dimension Designer Browser toolbar (highlighted in Figure 5.33) or choose Member Properties from the Dimension menu. A dialog appears that has all the attributes of the dimension that participate in attribute relationships. Select the attributes English Country Region Name, State Province Name, and City. Click OK. The member properties you have selected are now shown in the Dimension Browser, as shown in Figure 5.33.

Figure 5.33



Expand the members of United States to see the member properties of the States and Cities under United States. The member properties of a member are also retrieved with the help of an MDX query. For example, when you want to see all the cities in Alabama, the following MDX query is sent to the server:

```
WITH MEMBER [Measures].[-DimBrowseLevelKey 0-] AS
 '[Dim Geography].[Geography].currentmember.properties("key0", TYPED)'
MEMBER [Measures].[-DimBrowseLevelKey 1-] AS
 '[Dim Geography].[Geography].currentmember.properties("key1", TYPED)'

SELECT { [Measures].[-DimBrowseLevelKey 0-],
 [Measures].[-DimBrowseLevelKey 1-] ON 0,
 Head( [Dim Geography].[Geography].[State Province].&[Alabama].Children,
 1000) ON 1
FROM [$Dim Geography]
CELL PROPERTIES VALUE
```

Similar to the MDX query you analyzed earlier to retrieve the members of the All level, this query retrieves all the Alabama City members. The City attribute's member property State Province Name is retrieved (along with the values that make up the City attribute's composite key: City, and State Province Code) with the same query as calculated members using the `WITH MEMBER` clause as seen in the preceding query.

Sorting Members of a Level

Members of a level are the members of the attribute that makes up that level. For example, the members of the level Country in the Geography hierarchy are actually the members of the attribute English Country Region Name. The member name that is shown in the Dimension Designer Browser is the text associated with the name of the country. It is not uncommon for dimension tables to have one column for the descriptive name and one column that is the key column of the table. You can use the descriptive name column to display the name of the attribute and the key column to sort the members of that attribute.

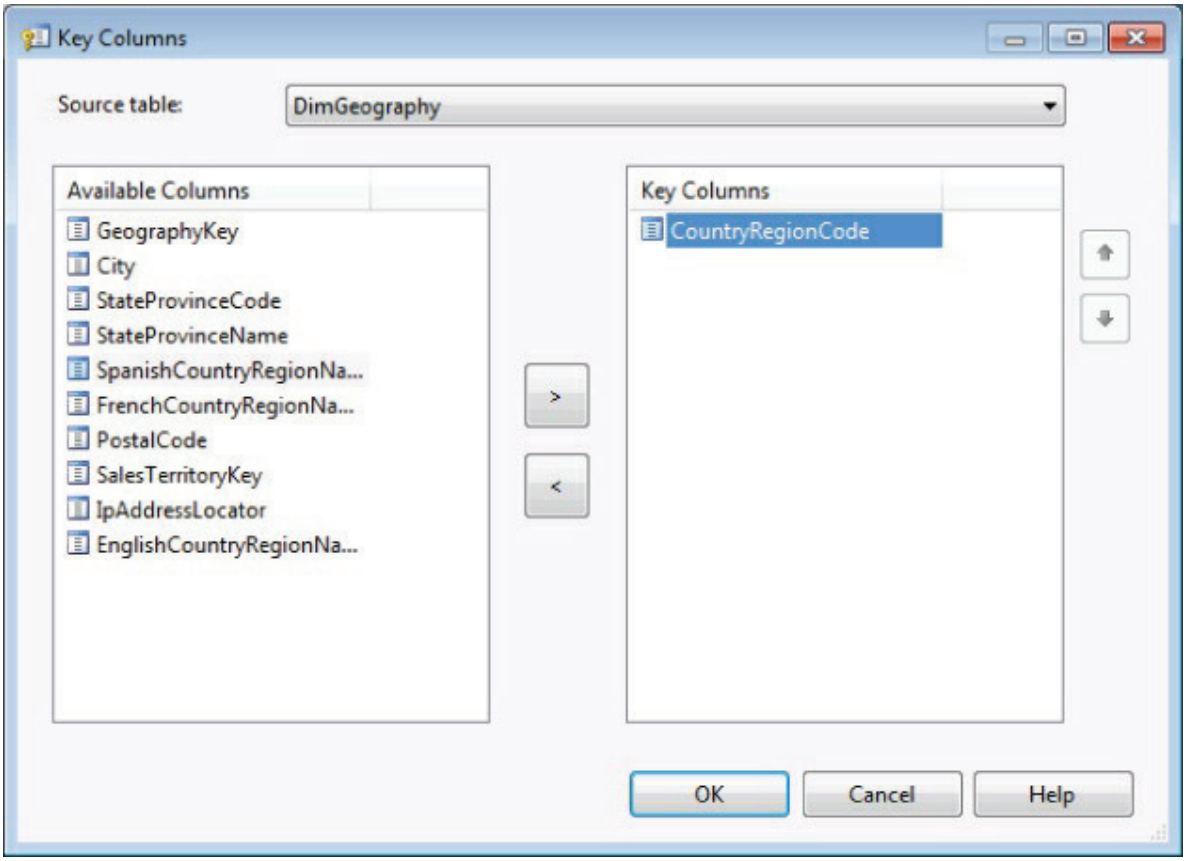
Each attribute in a dimension has the two properties: KeyColumns and NameColumn. The KeyColumns property is used to specify the columns that are used for sorting the members, and the NameColumn is used for the member's descriptive name. By default, the Dimension Wizard and the Dimension Designer set the KeyColumns property when an attribute is added to the dimension. They do not set the NameColumn property. If the NameColumn property is empty, Analysis Services returns the KeyColumns value for the member's descriptive name in response to client requests.

The KeyColumns property of the English Country Region Name attribute is set to be the column that the attribute itself is based on (EnglishCountryRegionName). Therefore, when you view the members in the Dimension Browser, they are sorted by the country names themselves. The Dim Geography dimension table also has the Country Region Code attribute. You can change the sort order of the countries to be based on the Country Region Code column instead of their names by changing the KeyColumns and NameColumn properties appropriately. The following exercise demonstrates how to do this.

1. In the Attributes pane, select English Country Region Name; then in the Properties pane, select the NameColumn property and click the Ellipsis button. This opens the Name Column dialog showing all the columns in the Dim Geography table. Select the EnglishCountryRegionName column, and click OK.

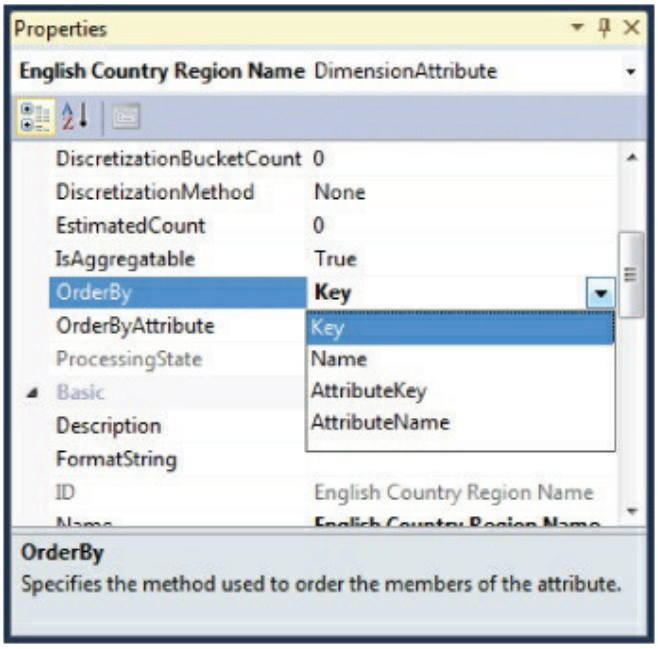
2. Select the KeyColumns property and click the Ellipsis button. This action launches the Key Columns dialog. Remove the EnglishCountryRegionName column from the collection. In the list of available columns, select CountryRegionCode and add it to the Key Columns list. The Key Columns selection dialog should look like Figure 5.34. Click the OK button.

Figure 5.34



3. In the Advanced section of the Properties window, make sure the value of the OrderBy property is Key, as shown in Figure 5.35. This instructs the server to order this attribute using the Key attribute (CountryRegionCode), which you specified in step 2.

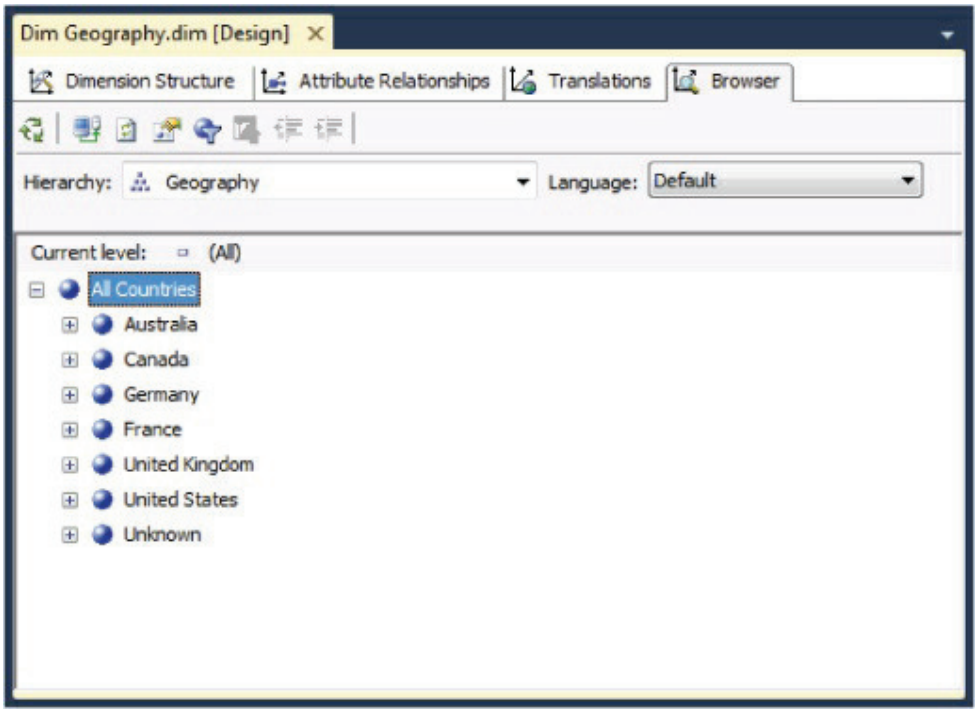
Figure 5.35



4. Deploy the project to the Analysis Services instance.

Deploying the project to the Analysis Services instance sends the new changes defined in steps 1 through 3 to the server and then processes the dimension. Switch to the Browser page, and click the Reconnect option to update the dimension data. In the Dimension Browser select the Geography hierarchy. The order of the countries has now changed to be based on the order of Country Region Code (AU, CA, DE, FR, GB, and US followed by the Unknown members). The new order of countries is shown in Figure 5.36.

Figure 5.36



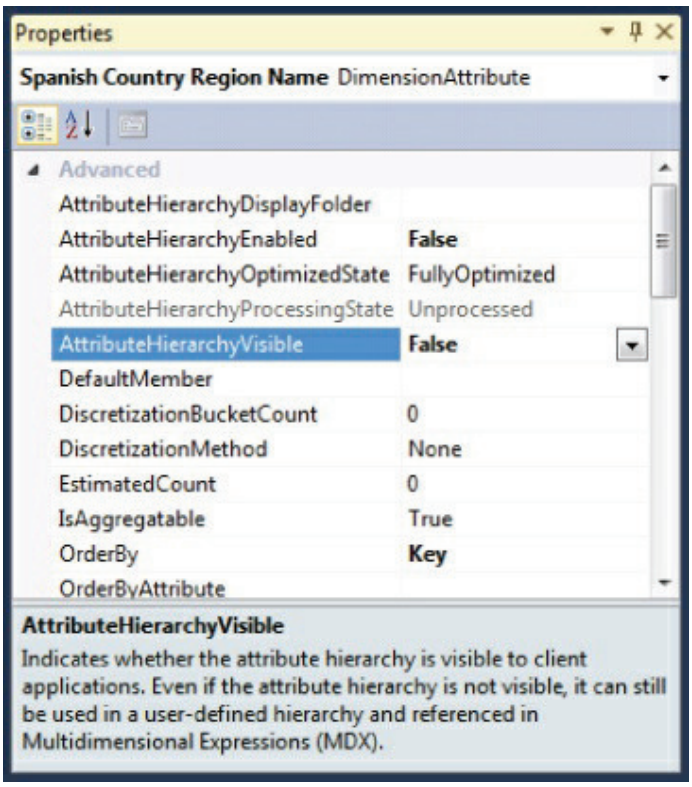
Optimizing Attributes

During the design of a dimension, you might want to include certain attributes but not have them available to end users for querying. Two attribute properties enable you to manipulate visibility of attributes to end users. One property, AttributeHierarchyEnabled, enables you to disable the attribute. By setting this property to False you can disable the attribute in the dimension; you cannot include this attribute in any level of a user hierarchy. This attribute can be defined only as a member property (related attribute) to another attribute. Members of this attribute cannot be retrieved by an MDX query, but you can retrieve the value as a member property of another attribute. If you

disable an attribute you might see improvements in processing performance depending on the number of members in the attribute. You need to be sure that there will be no future need to slice and dice on this attribute.

Another property, AttributeHierarchyVisible, is useful for making an attribute hierarchy invisible for browsing. Even with this set, however, the attribute can be used as a level within a user hierarchy, and it can be used for querying. If you set this property to False, you will not see this attribute in the Dimension Browser. The properties AttributeHierarchyEnabled and AttributeHierarchyVisible are part of the Advanced group in the Properties window, as shown in Figure 5.37.

Figure 5.37



If you want to create a dimension that contains only user hierarchies and no attribute hierarchies, you can set the AttributeHierarchyVisible property to False for all the attributes. When you go to the Dimension Browser you only see the multilevel hierarchies. Even though you have disabled the attributes for browsing, you can still query them using MDX.

Defining Translations in Dimensions

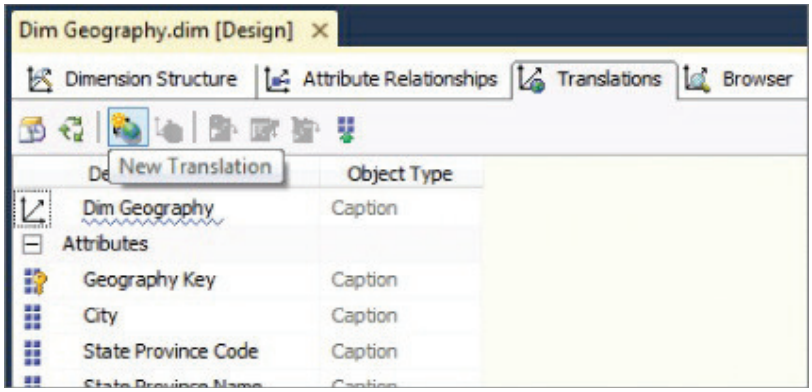
If your data warehouse is to be used globally, you may want to show the hierarchies, levels, and members in the languages that customers in those countries are familiar with. Analysis Services provides you with a feature called Translations that helps you create and view dimension members in multiple different languages. The benefit of this feature is that you do not have to build a new cube for every language you want to support if the only difference in functionality is the language of the data and metadata. For the translation feature to be used, you need only have columns in the relational data source that contain the translated member values for those attributes in the dimension whose values you would like to appear in the wanted languages.

For example, the Dim Geography table has two columns, Spanish Country Region Name and French Country Region Name, which contain the translated names of the countries that are members of the English Country Region Name attribute. The following steps describe how to create new translations for those languages:

1. Switch to the Translations page in the Dimension Designer.

2. Click the New Translation toolbar button, as shown in Figure 5.38, or choose New Translation from the Dimension menu to create a new translation and choose a language. The Select Language dialog appears.

Figure 5.38

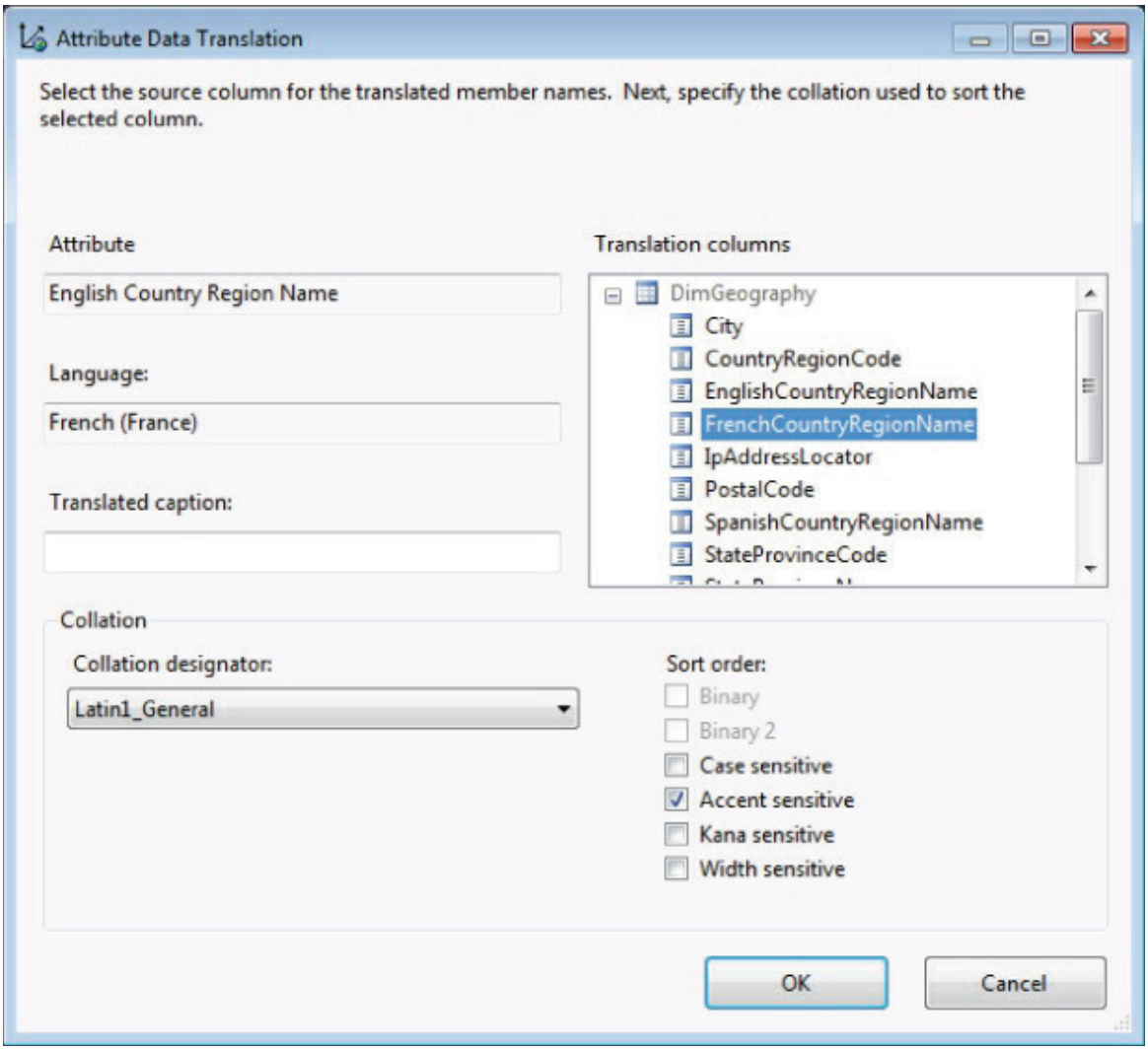


3. Select the language French (France) and click OK.

4. A new column with the title French (France) is added. Select the cell from the French (France) column in the English Country Region Name row. Then click the button that appears on the right side of the cell. You now see the Attribute Data Translation dialog.

5. Select the French Country Region Name column in the Translation Columns Tree view, as shown in Figure 5.39, and click OK.

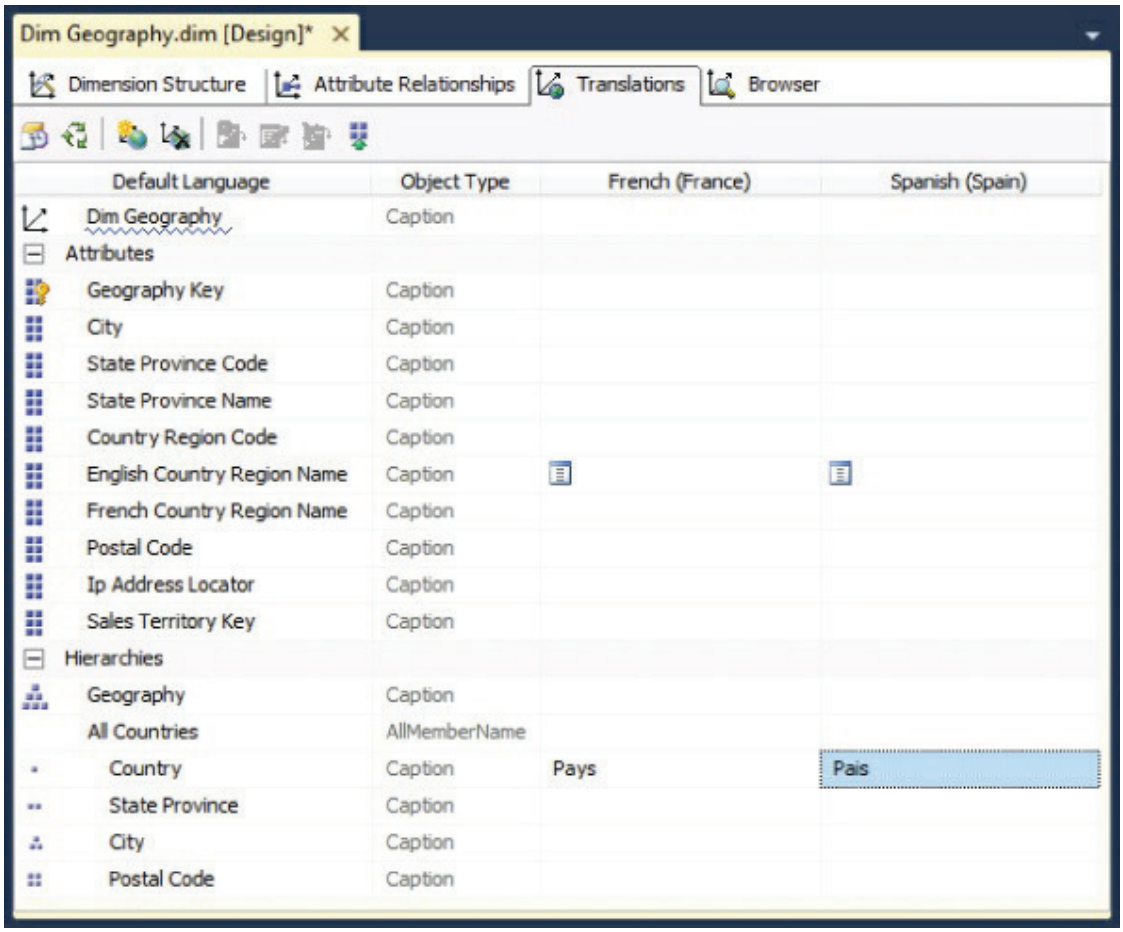
Figure 5.39



6. Repeat steps 2 through 5 for the Spanish (Spain) translation.

You have now created translations for the French and Spanish languages. In addition to specifying the columns for member names, you can also change the metadata information of each level. For example, if you want to change the name of the Country level in the Geography hierarchy in the French and Spanish languages, you can enter the names in the row that shows the Country level. Type **Pays** and **Pais**, as shown in Figure 5.40, for French and Spanish translations, respectively.

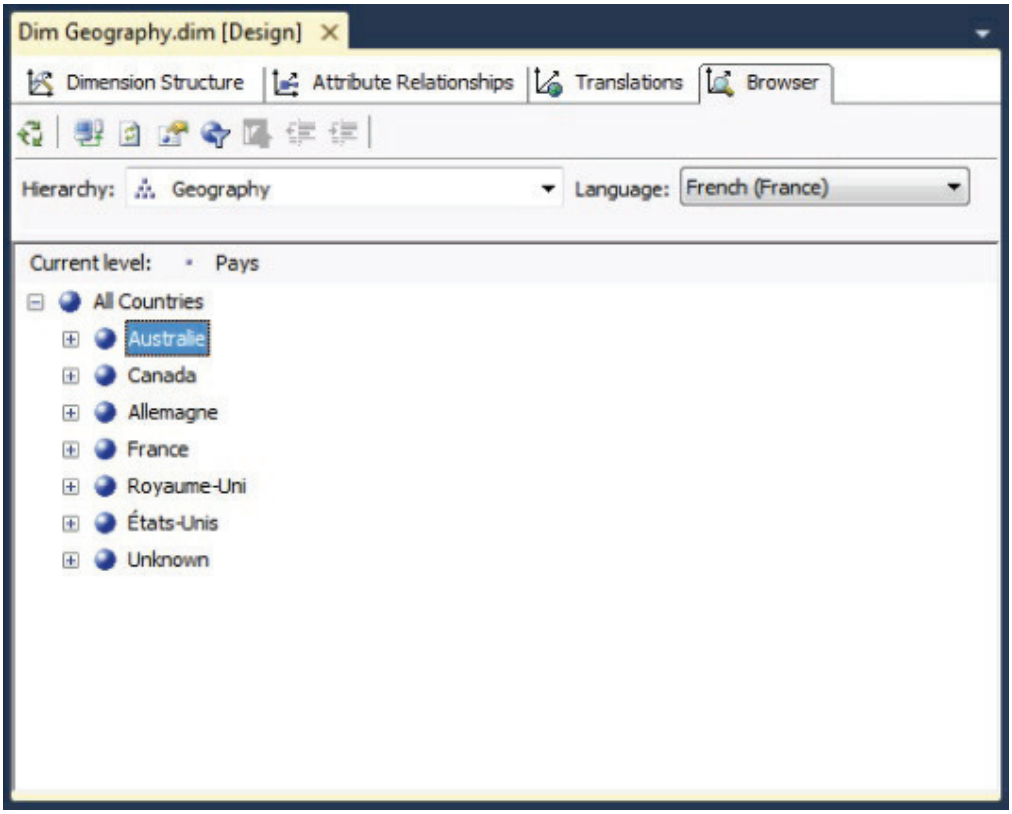
Figure 5.40



You have now defined translations for the Country attribute in two languages, making use of the columns in the relational data source. To see how this metadata information is shown in the Dimension Browser, first deploy the project to your Analysis Services instance.

Next, to see the translations you have created, select French (France) from the Language drop-down in the Dimension Browser's toolbar, as shown in Figure 5.41. Select the Geography hierarchy, and expand the All level. Now you can see all the members in French. If you click any of the countries, the metadata shown for the level is Pays (French for country). There is a negligible amount of overhead associated with viewing dimension hierarchies, levels, and members in different languages.

Figure 5.41



Creating a Snowflake Dimension

A snowflake dimension is a dimension that is created using multiple dimension tables. A snowflake dimension normally suggests that the tables in the data source have been normalized. Normalization is the process by which tables of a relational database are designed to remove redundancy and are optimized for frequent updates. Most database design books, including *The Data Warehouse Toolkit* (*Second Edition*) by Ralph Kimball (Wiley, 2002) and *An Introduction to Database Systems* by C. J. Date (Addison Wesley, 2003), talk about the normalization process in detail.

The columns from different tables of a snowflake dimension often result in levels of a hierarchy in the dimension. The best way to understand a snowflake dimension is to create one yourself. To do this you need to add two additional tables to your DSV. Following is how to add the two tables:

1. Open the Adventure Works DW DSV, and click the Add/Remove Objects button (top-left button in the DSV Designer toolbar).

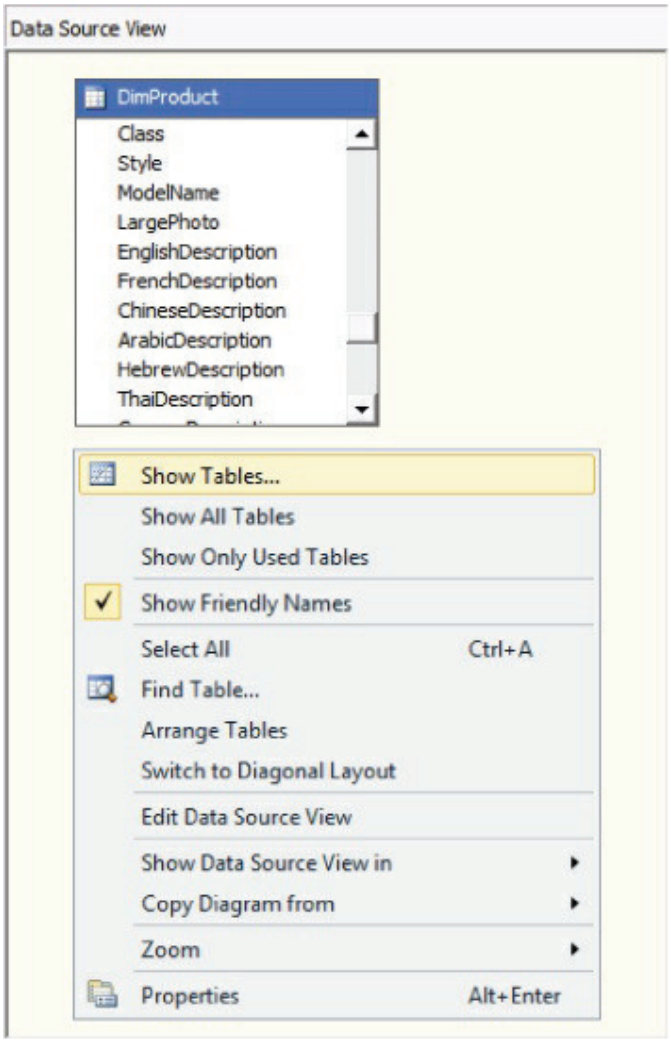
2. In the Available objects list of the Add/Remove Tables dialog, click DimProductCategory, Ctrl + Click DimProductSubcategory, and then the right arrow (>) button to move the two tables to the Included objects list. Click OK.

The DSV Designer identifies the relationships defined in the relational backend and shows the relationships between the DimProduct, DimProductSubcategory, and DimProductCategory tables within the DSV Designer's graphical design pane. Now that you have the necessary tables in the DSV with the relationships and primary keys defined, you can create a snowflake dimension using the DimProduct, DimProductCategory, and DimProductSubcategory tables. You can either delete the existing Product dimension in the AnalysisServicesMultidimensionalTutorial project and create a snowflake dimension using the Dimension Wizard, or refine the existing Product dimension to make it a snowflake dimension. In this example you use the latter approach. Follow these steps:

1. In the Solution Explorer, double-click the Dim Product dimension to open it in the Dimension Designer.

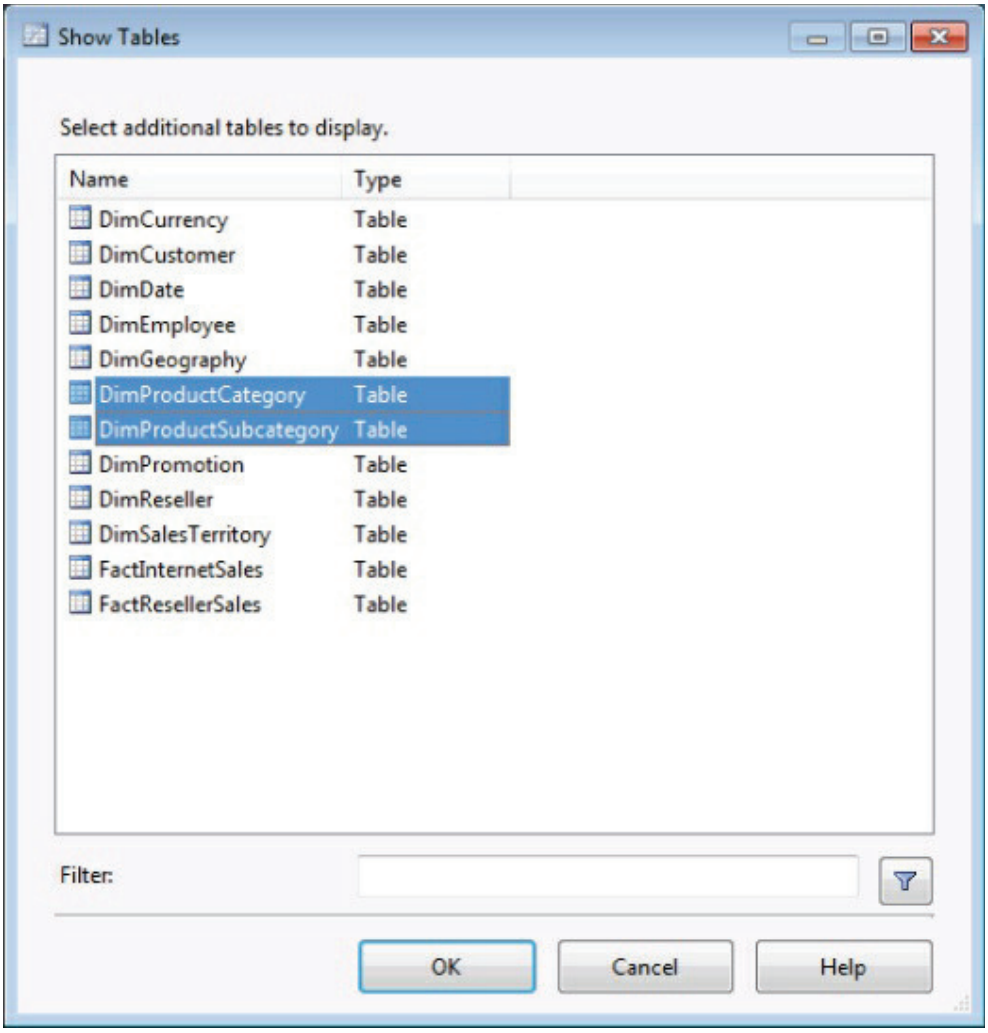
2. Within the Data Source View pane of the Dimension Designer, right-click and select Show Tables, as shown in Figure 5.42.

Figure 5.42



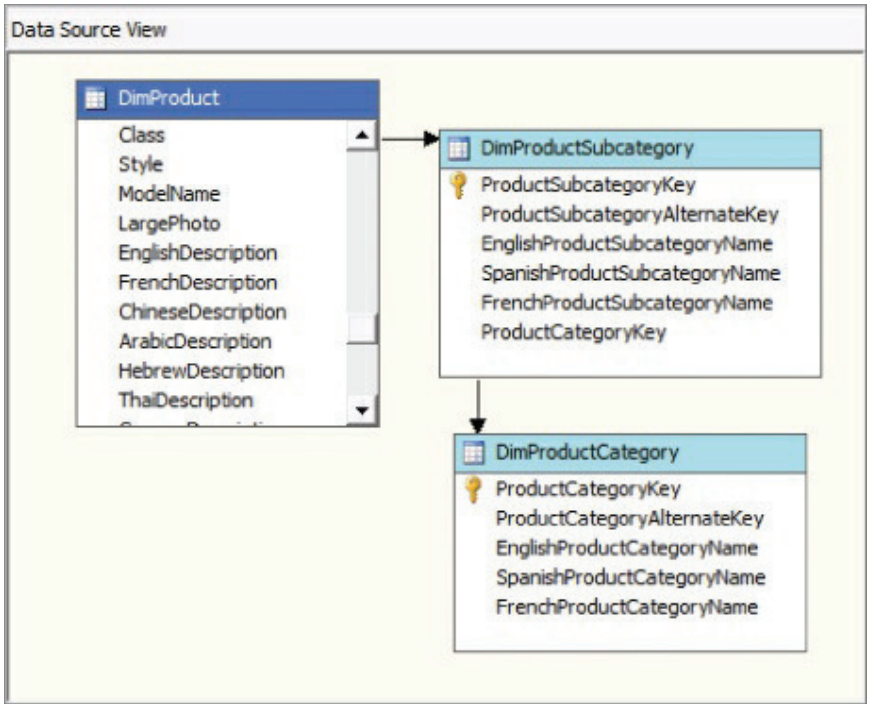
3. In the Show Tables dialog, select the DimProductCategory and DimProductSubcategory tables, as shown in Figure 5.43, and click OK.

Figure 5.43



You now see the DimProductSubcategory and DimProductCategory tables added to the Data Source View pane of the Dimension Designer, as shown in Figure 5.44. The new tables added to the pane have a lighter colored caption bar. This indicates that none of the columns in the tables are included as attributes of the dimension.

Figure 5.44



4. Drag and drop the ProductCategoryKey column from the DimProductSubcategory table in the DSV pane to the Attributes pane.

5. Launch the Name Column dialog for the Product Category Key attribute by clicking the ellipsis next to the NameColumn property in the Properties window.

6. Select DimProductCategory as the Source table, and then select EnglishProductCategoryName as the Source column. Click OK.

7. Select the Product Subcategory Key attribute in the Attributes pane.

8. Launch the Name Column dialog for the Product Subcategory Key attribute by clicking the ellipsis next to the NameColumn property in the Properties window.

9. Select DimProductSubcategory as the Source table, and then select EnglishProductSubcategoryName as the Source column; then click OK.

10. Launch the Name Column dialog for Product (the key attribute) by selecting it in the Attributes pane and clicking the ellipsis next to the NameColumn property in the Properties window.

11. Select DimProduct as the Source table and EnglishProductName as the Source column and click OK.

12. Create a new user hierarchy with levels Product Category Key, Product Subcategory Key, and Product by dragging and dropping the attributes to the Hierarchies pane and renaming the hierarchy Product Categories.

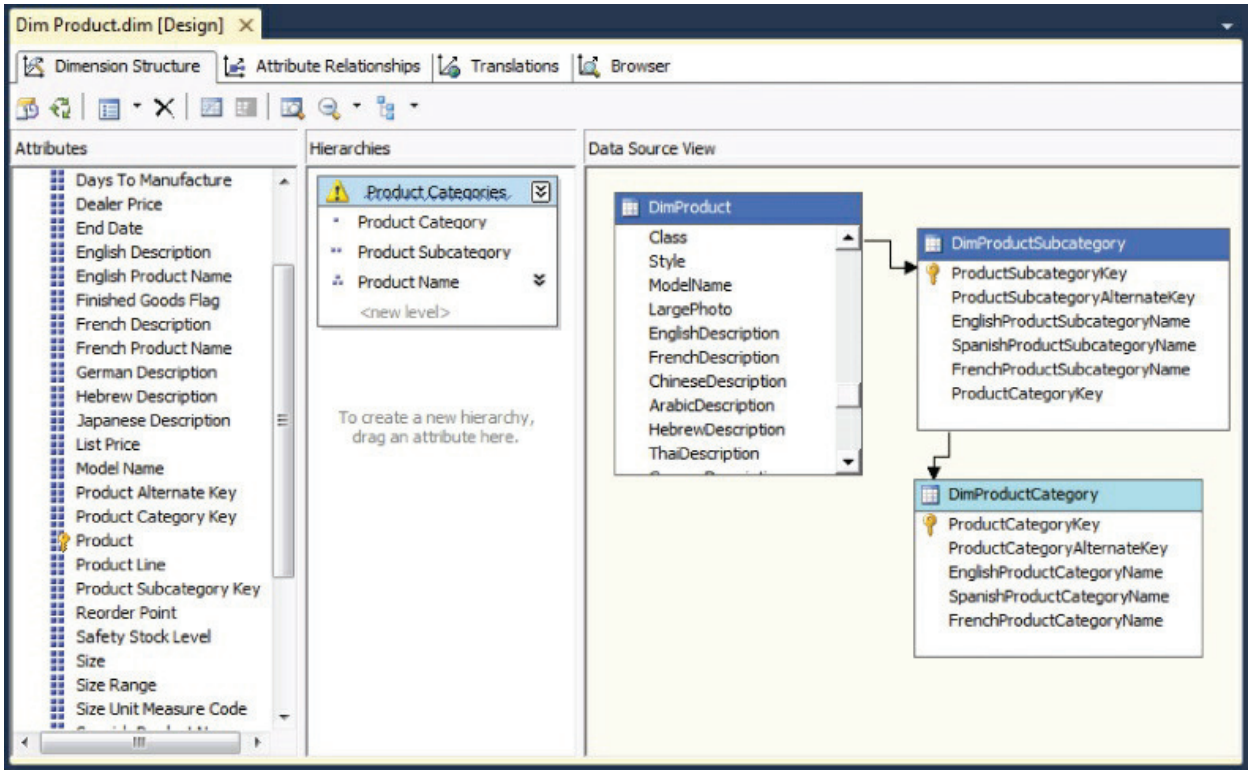
13. Rename the Product Category Key level Product Category.

14. Rename the Product Subcategory Key level Product Subcategory.

15. Rename the Product level Product Name.

16. Figure 5.45 shows the Product dimension in the Dimension Designer after all the preceding changes.

Figure 5.45



You have now successfully transformed the Dim Product dimension to use a snowflake schema. You can perform most of the same operations in a snowflake schema dimension as you can in a star schema dimension, including adding attributes, creating hierarchies, and defining member properties. We recommend you deploy the AnalysisServicesMultidimensionalTutorial project and browse the Dim Product snowflake dimension.

Creating a Time Dimension

Almost every data warehouse will have a Time dimension. The Time dimension can be composed of the levels Year, Semester, Quarter, Month, Week, Date, Hour, Minute, and Second. Most data warehouses contain the levels Year, Quarter, Month, and Date. The Time dimension allows analyzing business data across similar time periods — for example, determining how the current revenues or profit of a company compare to those of the previous year or previous quarter.

Even though it appears that the Time dimension has regular time periods, irregularities often exist. The number of days in a month varies, and the number of days in a year changes each leap year. In addition, a company can have its own fiscal year, which might not be identical to the calendar year. Even though there are minor differences in the levels, the Time dimension is often viewed as having regular time intervals. Several MDX functions help in solving typical problems related to analyzing data across time periods. ParallelPeriod is one such function, which you learned about in Chapter 3. Time dimensions are treated specially by Analysis Services, and certain measures are aggregated across the Time dimension uniquely. Measures aggregated this way are called semi-additive measures. You learn more about semi-additive measures in Chapters 6 and 9.

The AnalysisServicesMultidimensionalTutorial project has a time dimension called Dim Date that was created by the Cube Wizard in Chapter 2. Even though the dimension has been created from the Dim Date table, it does not have certain properties set that would allow Analysis Services to see it as the source for a Time dimension. In the following exercise, you first delete the Dim Date dimension and then re-create it as a Time dimension. Follow these steps to create a Time dimension on the DimDate table of the AdventureWorksDW database:

1. In the Solution Explorer, right-click the Dim Date dimension, and select Delete.

2. In the Delete objects and files dialog, SSDT asks you to confirm the deletion of the corresponding Cube dimensions (you learn about Cube dimensions in Chapter 9). Select OK to delete the Dim Date dimension.

3. Launch the Dimension Wizard by right-clicking the Dimensions folder in the Solution Explorer and selecting New Dimension. When the Welcome screen of the Dimension Wizard opens, click Next.

4. In the Select Creation Method page of the wizard, select Use an Existing Table, and click Next.

5. In the Specify Source Information page, select DimDate as the main table from which the dimension is to be designed, and click Next.

6. In the Select Dimension Attributes page, in addition to the Date Key attribute, check the following attributes: Calendar Year, Calendar Semester, Calendar Quarter, English Month Name, and Day Number Of Month.

7. Set the Attribute Type for the Day Number of Month attribute to Date Í Calendar Í Day of Month, as shown in Figure 5.46.

Figure 5.46

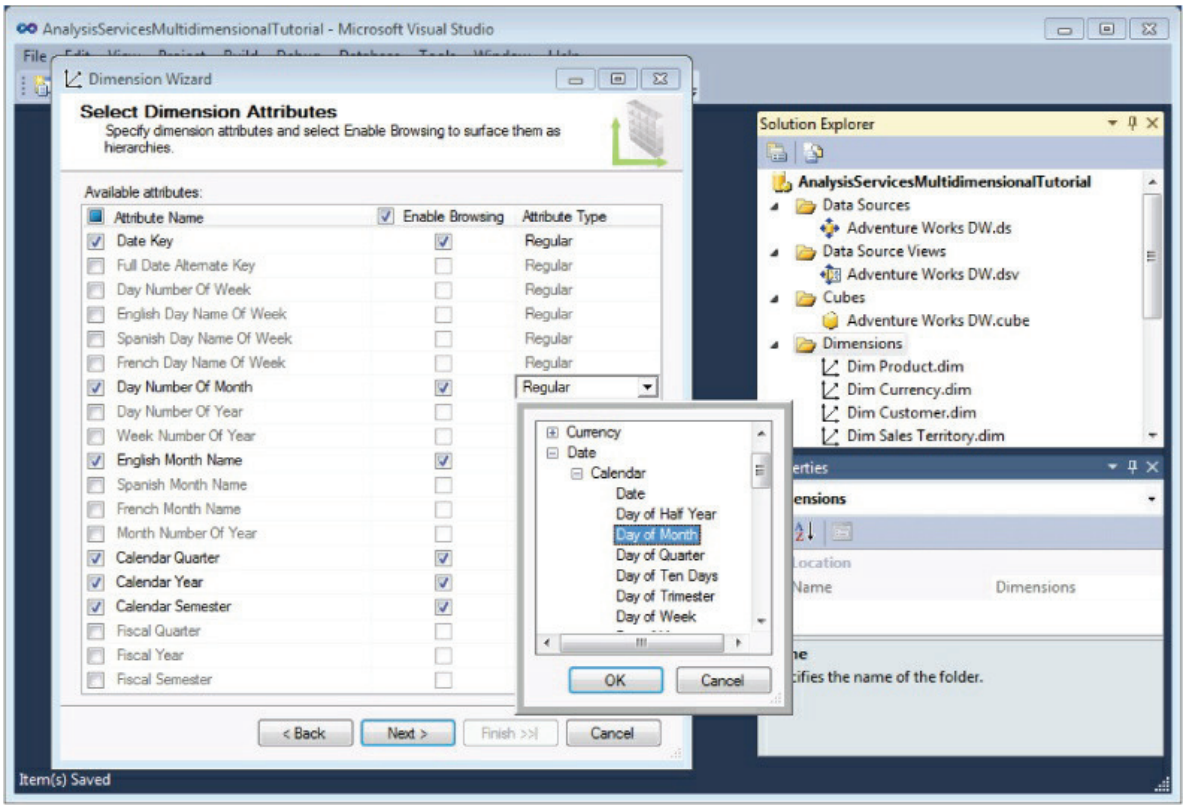
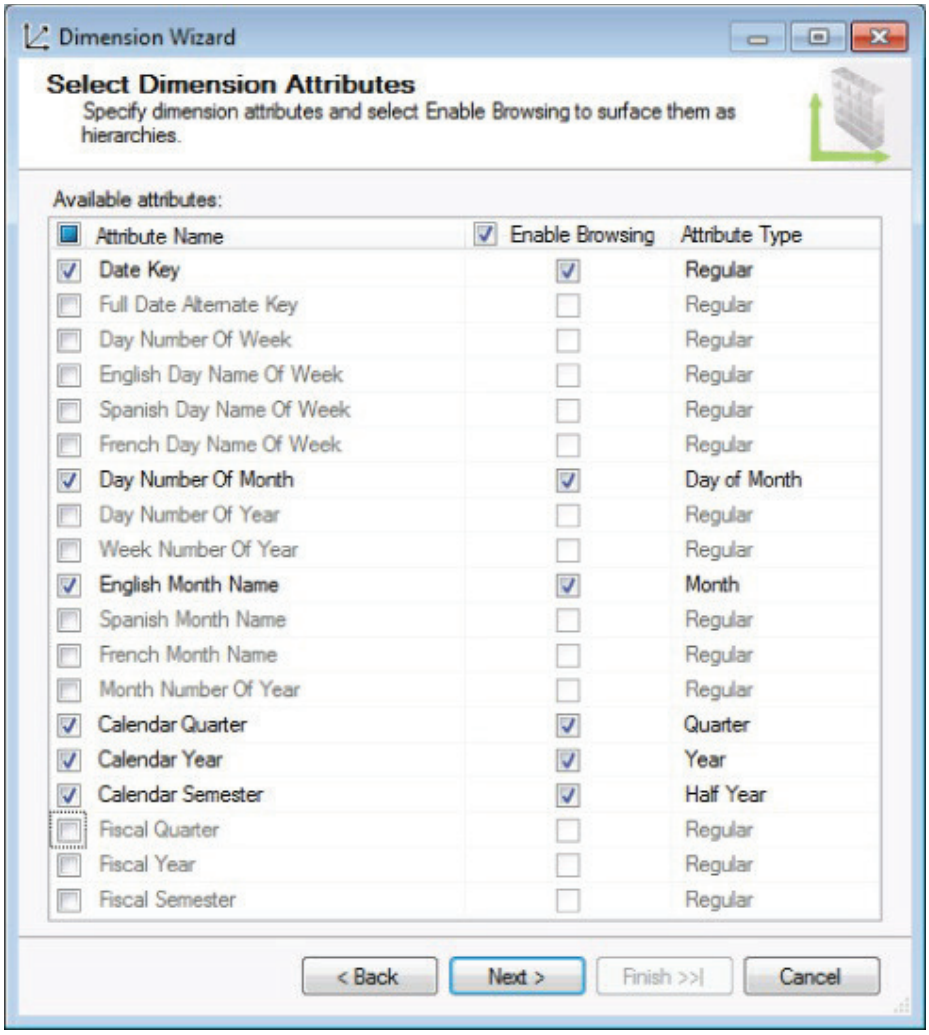


Figure 5.47



8. Set the Attribute Type for the remaining enabled attributes so they match those shown in Figure 5-47, and click Next to continue.

9. Set the name of the dimension to Dim Date, and click Finish to close the Dimension Wizard. You have now successfully created a Time dimension using the Dimension Wizard.

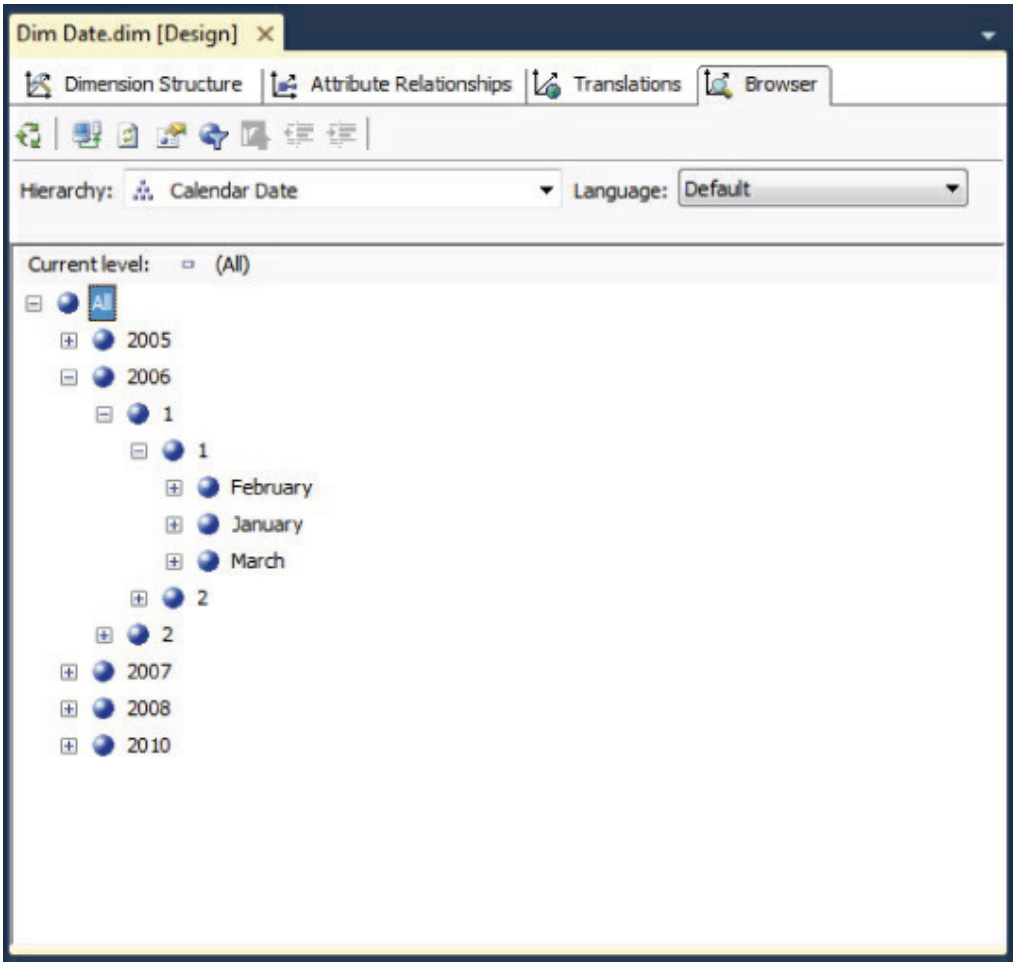
10. Create a multilevel hierarchy Calendar Date with the levels Calendar Year, Calendar Semester, Calendar Quarter, Month (rename from English Month Name), and Day (rename from Day Number of Month).

11. Save the project and deploy it to the Analysis Services instance.

12. Switch to the Browser pane of the Dim Date dimension.

Figure 5.48 shows the Calendar Date hierarchy that you created. Notice that the order of months within a quarter is not the default calendar order. For example, the order of months of CY Q1 of year 2006 is February, January, and March. To change the order, change the KeyColumns, NameColumn, and OrderBy properties appropriately and redeploy the project. Define the necessary attribute relationships and attribute key values as defined by your business needs.

Figure 5.48



You have now successfully created a Time dimension. If you click the Dim Date dimension in the Attributes pane of the Dimension Structure page of the designer and review the properties of the Dim Date dimension, you see the Type property set to Time that indicates that the Dim Date dimension is a Time dimension. If you review the basic properties of each attribute in the Dim Date dimension, you notice that the Type property has values such as Quarter, Half Year, Year, Day Of Month, and Month. Setting the right Type property is important because client applications can use this property to apply the appropriate MDX functions for a Time dimension.

Creating a Parent-Child Hierarchy

In the real world you come across relationships such as that between managers and their direct reports. This is similar to the relationship between a parent and child in that a parent can have several children, and a parent can also be a child because parents also have parents. In the data warehousing world, such relationships are modeled as Parent-Child dimensions, and in Analysis Services this type of relationship is modeled as a hierarchy called a Parent-Child hierarchy. The key difference between this relationship and any other multilevel hierarchy is how this relationship is represented in the data source. Well, that and certain other properties that are unique to the Parent-Child design. Both of these are discussed in this section.

When you created the Geography dimension, you might have noticed that there were separate columns for Country, State, and City in the relational table. Similarly, the manager and direct report can be modeled by two columns, ManagerName and EmployeeName, where the EmployeeName column is used for the direct report. If there were five direct reports for a manager, there would be five rows in the relational table. The interesting part of the Manager-DirectReport relationship is that the manager is also an employee and is a direct report to another manager. This is unlike the City, State, and Country columns in the Dim Geography table.

It is probably rare at your company, but employees can sometimes have new managers due to reorganizations. The fact that an employee's manager can change at any time is interesting when you want to look at facts such as sales generated under a specific manager, which is the sum of sales generated by the manager's direct reports. A dimension modeling such a behavior is called a *slowly changing dimension* because the manager of an employee changes over time. You can learn about slowly changing dimensions and different variations in detail in the book *The Microsoft Data Warehouse Toolkit*: *With SQL Server 2008 R2 and the Microsoft Business Intelligence Toolset* by Joy Mundy *et al.* (Wiley, 2011).

The DimEmployee table in the AdventureWorksDW relational database has a Parent-Child relationship because it has a join from the

ParentEmployeeKey column to the EmployeeKey column. You have already created a DimEmployee dimension in the AnalysisServicesMultidimensionalTutorial project in Chapter 2 using the Cube Wizard. In the following exercise you refine the existing Dim Employee dimension and learn how to create a dimension with a Parent-Child hierarchy using the Dimension Wizard. You can actually refine, not create, the Dim Employee dimension in the illustration.

1. Launch the Dimension Wizard by right-clicking the Dimensions folder in the Solution Explorer and selecting New Dimension. If the Welcome screen of the Dimension Wizard opens up, click Next.

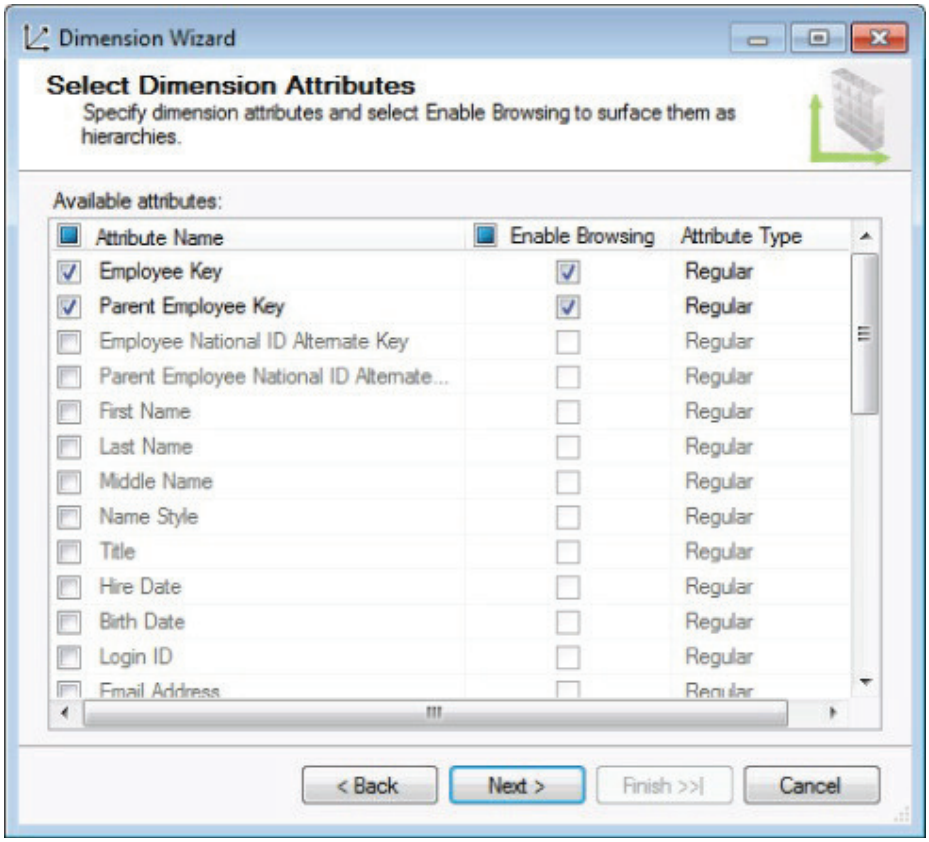
2. Make sure the Use an Existing Table option is selected, and click Next.

3. In the Specify Source Information page, select DimEmployee as the main table from which the dimension is to be designed, and click Next.

4. On the Select Related Tables screen, uncheck the DimSalesTerritory table, and click Next.

In the Select Dimension Attributes dialog, the Dimension Wizard has detected three columns of the DimEmployee table to be included as attributes. The Dimension Wizard selects columns if they are either part of the primary key of the table or a foreign key related to the primary key in the table or a foreign key related to the primary key in another table in the DSV. Figure 5.49 shows two of the attributes selected by the Dimension Wizard from the DimEmployee table. The attributes suggested by the Dimension Wizard in this example are the key attribute Employee Key, the parent-child attribute Parent Employee Key, and the Sales Territory Key, which is a foreign key column to the DimSalesTerritory table.

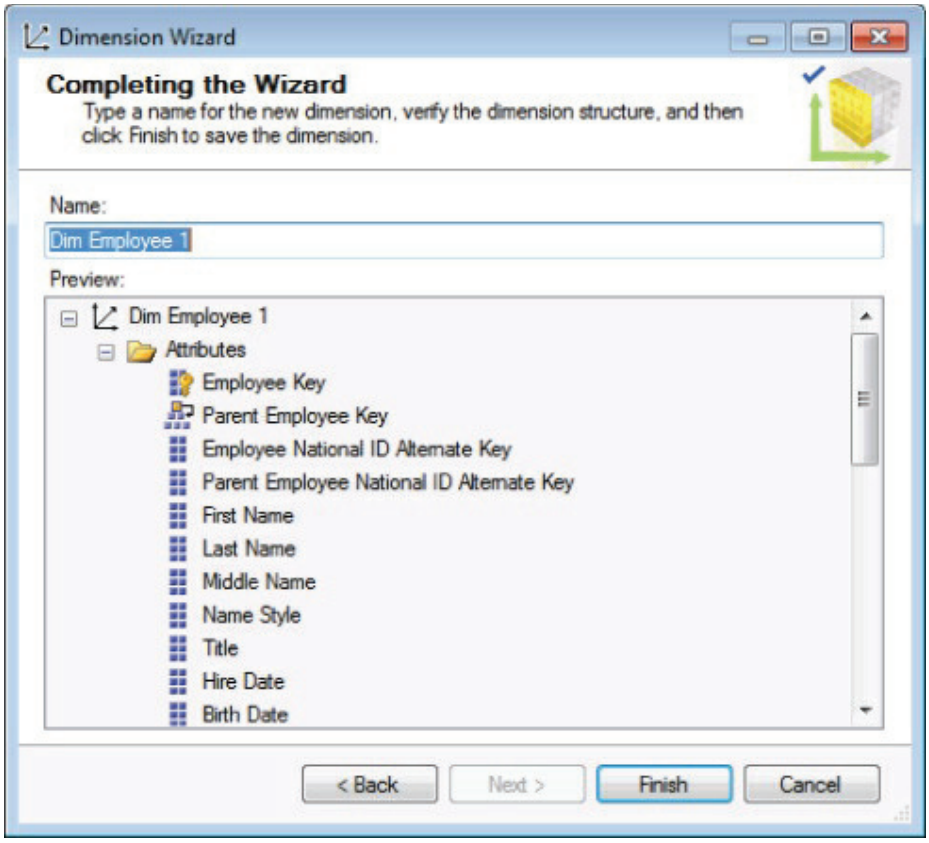
Figure 5.49



5. Select all the columns of the DimEmployee table as attributes, and click Next.

6. In the preview pane of the Completing the Wizard dialog, the Parent Employee Key attribute has a unique icon (see Figure 5.50) indicating that Analysis Services detected a Parent-Child relationship in the DimEmployee table. The wizard identified the Parent-Child relationship due to the join within the same table in the DSV.

Figure 5.50



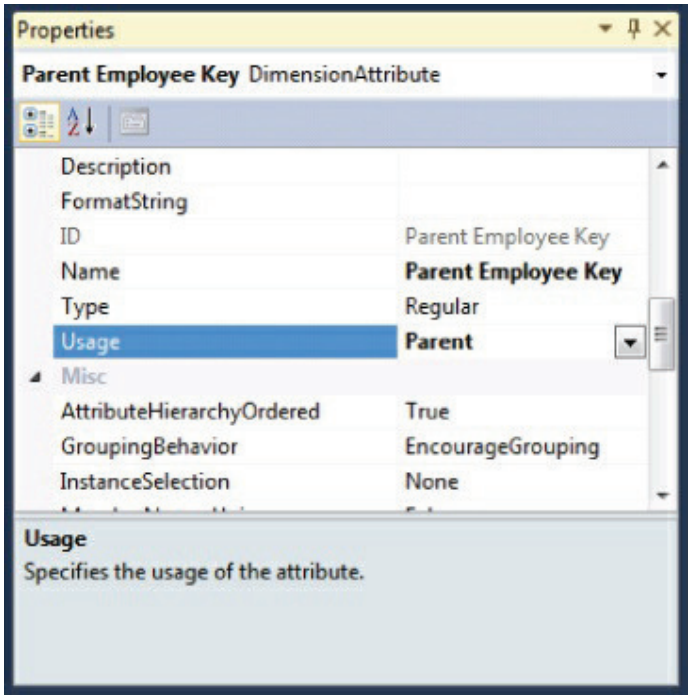
7. Click the Cancel button because you will not be creating another Dim Employee dimension.

By default the Dimension Wizard defines the properties for the attribute modeling the Parent-Child hierarchy at the completion of the Dimension Wizard or the Cube Wizard.

8. Double-click the Dim Employee dimension in the Solution Explorer to open it in the Dimension Designer.

9. The properties of the Parent Employee Key attribute indicate that this attribute defines a Parent-Child hierarchy, as shown in Figure 5.51.

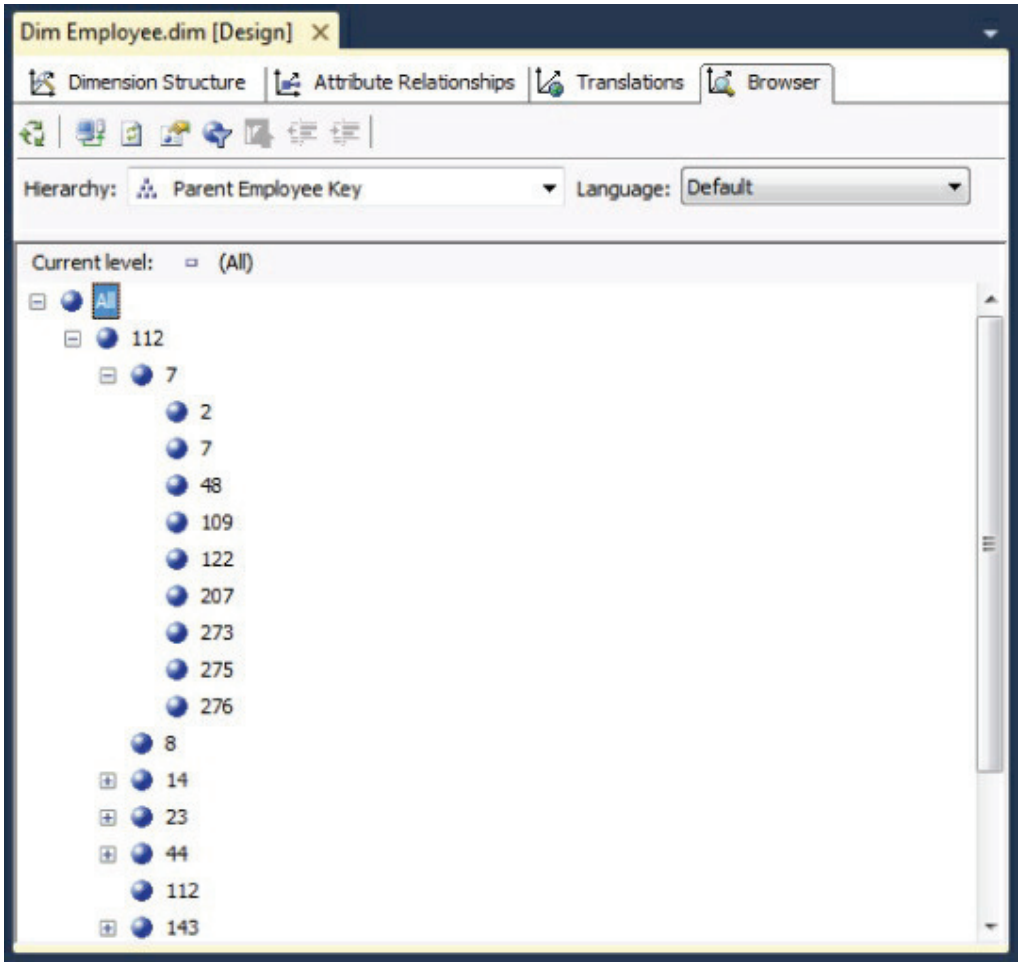
Figure 5.51



The hierarchy doesn't appear in the Hierarchies pane of the Dimension Designer. That's because the Parent-Child hierarchy is

actually a special type of attribute hierarchy that can contain multiple levels, unlike the other attributes. The Parent-Child hierarchy that the wizard created is based on the attribute ParentEmployeeKey. The Usage property for this attribute is set to Parent, which indicates that this attribute is a Parent-Child hierarchy. If you browse the Parent-Child hierarchy of the DimEmployee dimension, you can see the IDs of the employees as a multilevel hierarchy, as shown in Figure 5.52.

Figure 5.52

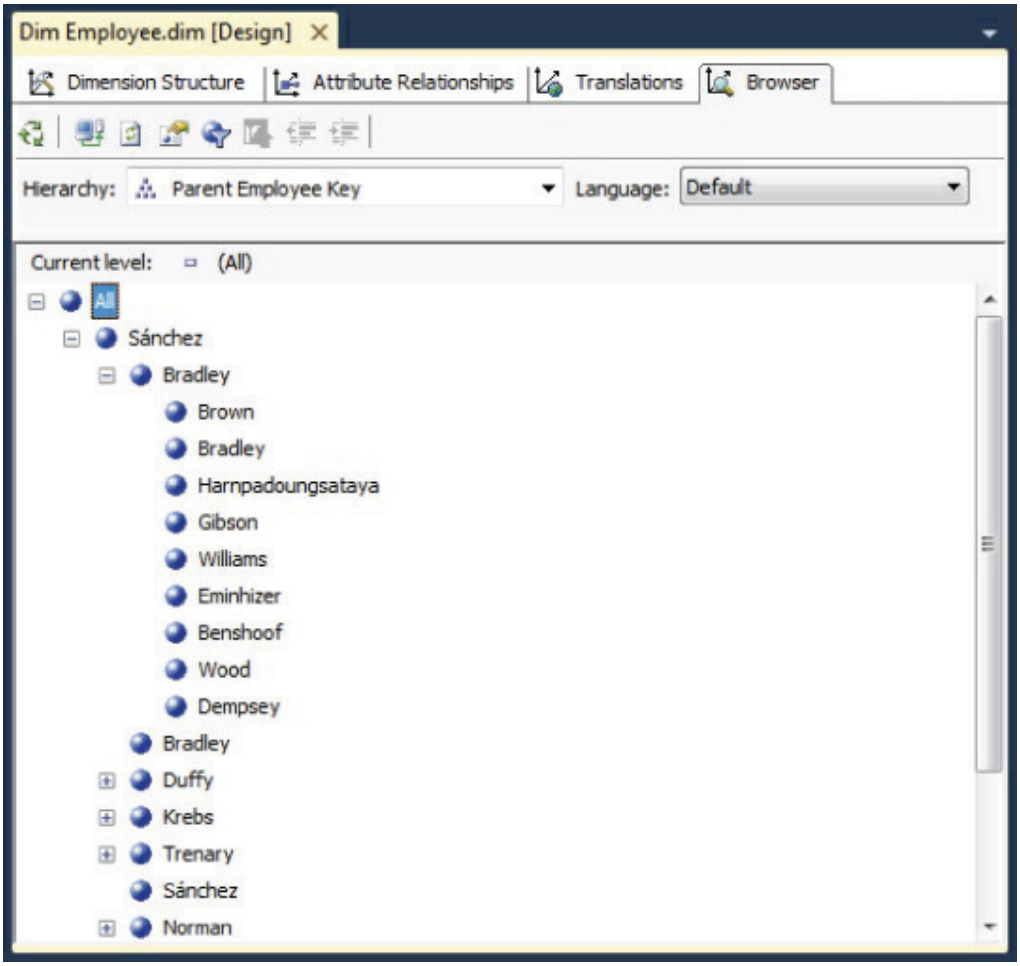


Typically, you would want to see the names of the employees rather than their IDs. You learned earlier that you can use the NameColumn property to specify the name that is shown in the browser and use the key column for ordering. Because the Parent-Child hierarchy retrieves all the information from the Key attribute, which is the Employee attribute in this example, you need to modify the NameColumn property of the Employee attribute rather than the NameColumn property of the Parent Employee Key attribute.

10. Change the NameColumn property of the Employee attribute to **LastName** and deploy the project to your Analysis Services instance.

When you browse the Parent-Child hierarchy, you see the members of the hierarchy showing the last names of the employees, as shown in Figure 5.53.

Figure 5.53



Summary

Using the Dimension Wizard and other wizards in SSDT is only the starting point for designing objects in Analysis Services. For optimal results, you need to take your dimensions farther than the initial destination that the wizards take you to. To get to the ultimate destination of your dimension design, use the Dimension Designer you learned about in this chapter. A couple of examples of steps along this last leg of the journey are using the Properties window to assign descriptive names to an attribute whose name in the source database might be obscure to your end users and defining attribute relationships to optimize dimension performance. You can also use the Dimension Designer to create translations for the attributes and hierarchies of a dimension to enhance the user experience of your international customers.

In addition to learning about dimensions, you learned the necessity of deploying your dimension definition to the instance of Analysis Services where the dimension is processed by retrieving the data from the data source. Processing is essential to enable you to browse a dimension. The communication between SSDT and an instance of Analysis Services is accomplished through a SOAP-based XML API called XML for Analysis (XMLA), which is an industry standard. Even more interesting, dimensions stored in Analysis Services are represented internally as cubes — one-dimensional cubes; and, coincidentally, cubes are the topic of Chapter 6.

Chapter 6

Cube Design

What's in this chapter?

- Understanding the BISM multidimensional mode
- Creating cubes with the Cube Wizard
- Browsing cubes with the Cube Browser and Excel
- Understanding cube dimensions and dimension relationships
- Learning measure and Measure Group properties
- Creating calculated members and measures
- Creating and browsing perspectives and translations

In Chapter 5 you learned to create dimensions using the Dimension Wizard and to refine and enhance dimensions using the Dimension Designer. Dimensions by themselves aren't that useful; you need to incorporate them into a cube to fulfill their purpose: enabling users to slice-and-dice data to gain insights. In this chapter you learn how to create cubes using the Cube Wizard and enhance your cubes using the Cube Designer. You learn to add calculations to your cube that facilitate effective data analyses followed by analyzing the cube data itself in the Cube Designer and Excel.

The BISM Multidimensional mode

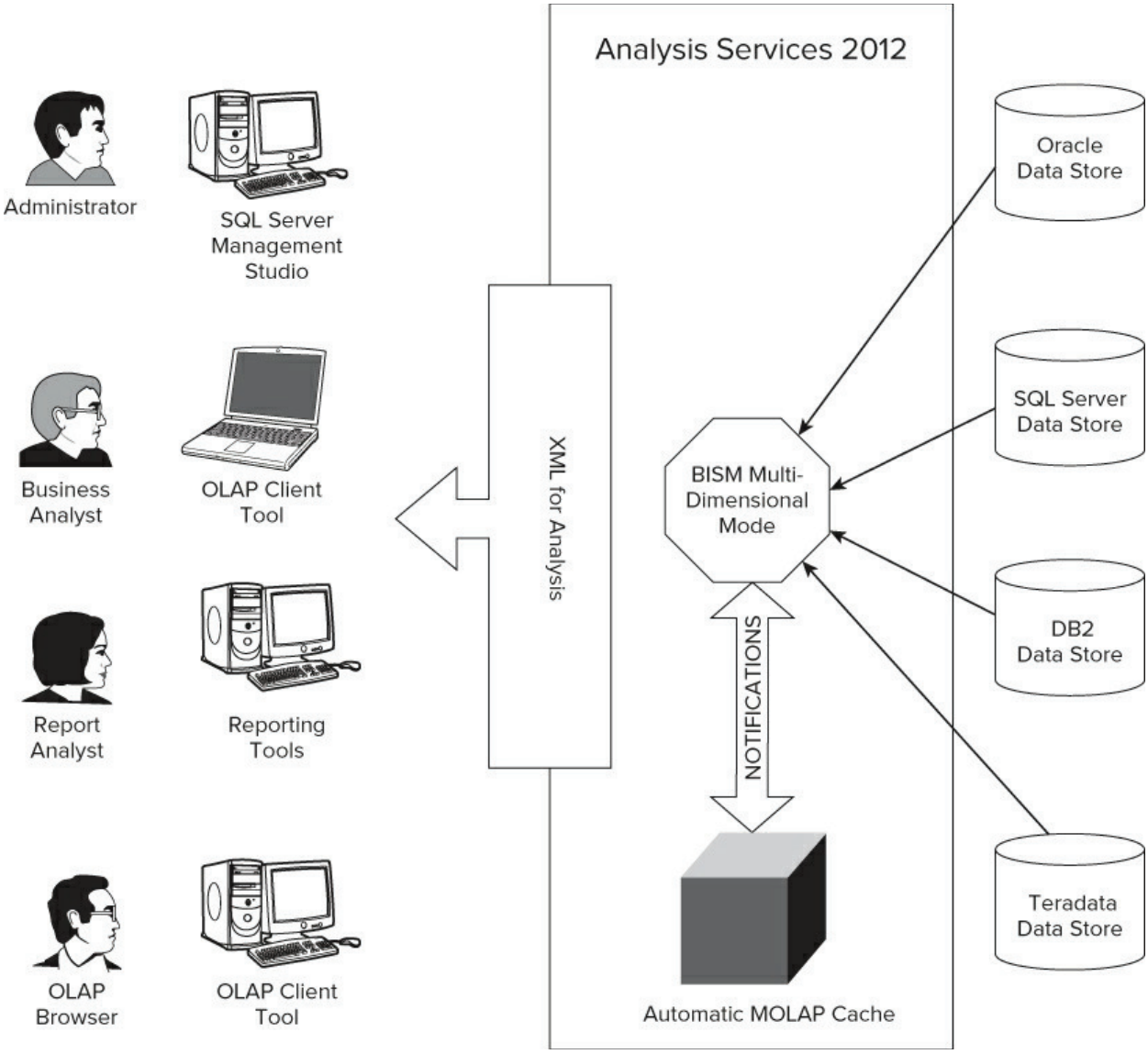
To generate profits for a business, you need to make key strategic decisions based on likely factors such as having the right business model, targeting the right consumer group, pricing the product correctly, and marketing through optimal channels. To make the right decisions and achieve targeted growth you need to analyze data. The data can be past sales, expected sales, or even information from competitors. The phrase “knowledge is power” is fitting here because in the world of business, analyzing and comparing, for example, current sales against the expected sales, helps executives make decisions directly aligned with the goals of the company. Such sales information is typically stored in a distributed fashion and must be collected from various sources. Executives making the business decisions typically do not have the ability to access the raw sales data spread across various locations and subsequently optimize it for their use. These decision-makers typically rely on the data that has already been aggregated into a form that is easy to understand and that facilitates the decision-making process. Presenting aggregated data to the decision-makers quickly is a key challenge for business intelligence providers. Analysis Services enables you to design a model that bridges the gap between the raw data and the information content that can be used for making business decisions. This model is called the Business Intelligence Semantic Model (BISM).

In this section you work with the multidimensional mode of the BISM. Later chapters deal with the other mode of the BISM: the tabular mode.

The BISM multidimensional mode enables you to bring data from multiple heterogeneous sources into a single model. Analysis Services buffers you from the difficulties of managing the integration of various data sources, so you can build your model easily. It provides you with the best of the OLAP and relational worlds, exposing rich data and metadata for exploration and analysis.

Figure 6.1 shows the architecture of the multidimensional model implemented in Analysis Services 2012. As shown in the figure, the multidimensional model helps you integrate data from various relational data sources — such as Oracle, SQL Server, DB2, and Teradata — as well as flat files into a single model that merges the underlying schemas into a single schema. The end users do not necessarily have to view the entire schema of the model. Instead, they can view the sections of it that are relevant to their needs through a feature provided by Analysis Services called perspectives.

Figure 6.1



In the OLAP world, data analyzed by end users is often historical data that might be a few days, months, or even years old. However, the responses to OLAP queries are typically returned within a few seconds. In the relational world the end users have instant access to the raw data, but the responses to queries can take much longer, on the order of minutes. As mentioned earlier, the multidimensional mode merges the best of both the OLAP and relational worlds and provides end users with real-time data with the query performance of the OLAP world. The multidimensional mode can provide that query performance with the help of a feature in Analysis Services that creates a cache of the relational data that also aggregates the data into an Analysis Services database. During the time the cache is being built, Analysis Services retrieves the data directly from the data sources. As soon as the cache is available, the results are retrieved from the cache in response to relevant queries. Whenever there is a change in the underlying data source, Analysis Services receives a notification and appropriate updates are made to the cache based on the settings defined for cache updates.

The multidimensional mode also provides rich, high-end analytic support through which complex business calculations can be exploited. Such complex calculations can be extremely difficult to formulate in the relational world at the data-source level. Even if such calculations are defined on the relational data source, responses from OLAP-style queries against the relational data source are typically slow compared to responses from Analysis Services.

Analysis Services natively interfaces to end-user clients through the XML for Analysis (XMLA) standard, which allows client tools to retrieve data from Analysis Services. Client tools such as Excel allow end users to create ad-hoc queries for data analysis. In addition, the multidimensional mode supports rich analytic features such as Key Performance Indicators (KPIs), Actions, and Translations that help surface the status of your business data at any given time so that appropriate actions can be taken.

The multidimensional mode provides an efficient interface for detail-level reporting through dimension attributes that are common in the relational world. The ability to transform multidimensional results into views that are helpful to end users and the ability to perform ad-hoc queries on data from high-level aggregated data to detail-level items, make the multidimensional model a powerful construct indeed. The multidimensional mode also enables you to display the model's data and metadata in the end user's language, which is needed in a global market.

Creating a Cube Using the Cube Wizard

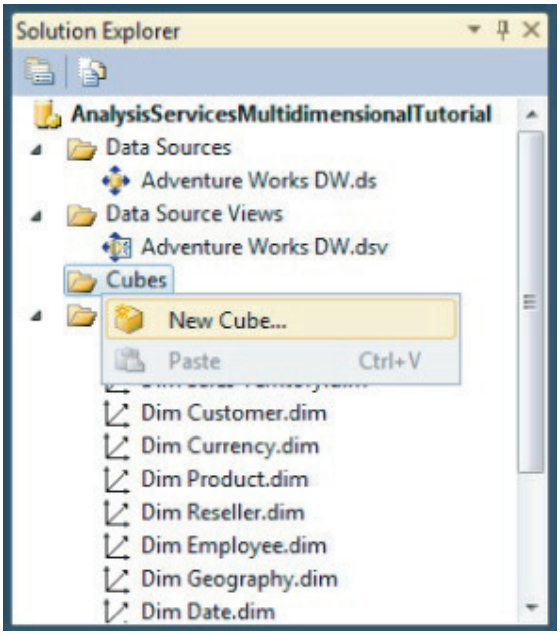
Cubes are the principal objects of an OLAP database and are multidimensional structures primarily composed of dimensions and facts. *Measures* are the fact table data stored in a cube. *Measure Groups* are groups of measures typically sourced from the same fact table and associated with the same set of dimensions. In Analysis Services you can store data from multiple fact tables within the same cube. Chapter 2 introduced you to the Cube Wizard. In this chapter you see more details of the Cube Wizard and also learn how to make refinements to your cube using the Cube Designer.

Similar to the Dimension Wizard you used in Chapter 5, the Cube Wizard facilitates creation of cube objects from the DSV. For this exercise, you continue with the AnalysisServicesMultidimensionalTutorial project you updated in Chapter 5, which contained the Dim Geography, Dim Employee, and Dim Date dimensions. To start with a clean slate, delete the existing Adventure Works DW cube if it is still there from Chapter 2. To completely understand the functionality of the Cube Wizard, follow these steps to build a new cube from scratch:

1. Open the AnalysisServicesMultidimensionalTutorial project from Chapter 5. If the Adventure Works DW cube exists, delete it by right-clicking it in the Solution Explorer and selecting Delete.

2. Right-click the Cubes folder and select New Cube, as shown in Figure 6.2. Click Next on the introduction page to proceed.

Figure 6.2

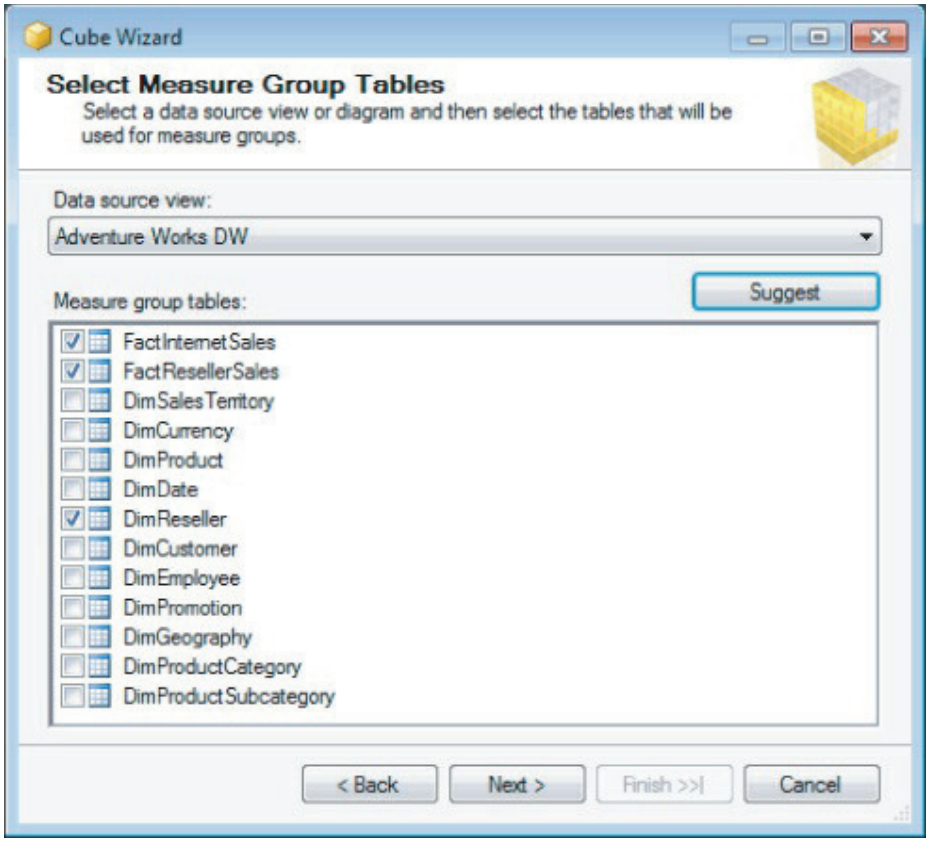


3. On the Select Creation Method page, you have the option to build a cube from existing tables, create an empty cube, or build a cube from tables you generate in the data source (optionally based on an existing template. In this tutorial you build the cube from the existing tables in the Adventure Works DW data source). Click Next to proceed to the next step in the Cube Wizard.

4. The next page of the Cube Wizard is the Select Measure Group Tables page. On this page, select the DSV the cube is based on and the tables in that DSV to serve as the fact tables for you measure group. Use the Suggest button on this page to have the Cube Wizard scan the DSV to detect the fact tables. Click the Suggest button to have the Cube Wizard automatically select potential Measure Group tables.

The Cube Wizard scans the DSV to detect the fact tables and automatically selects the candidate tables in the Measure group tables list, as shown in Figure 6.3. Any table that has an outgoing relationship is identified as a candidate fact table.

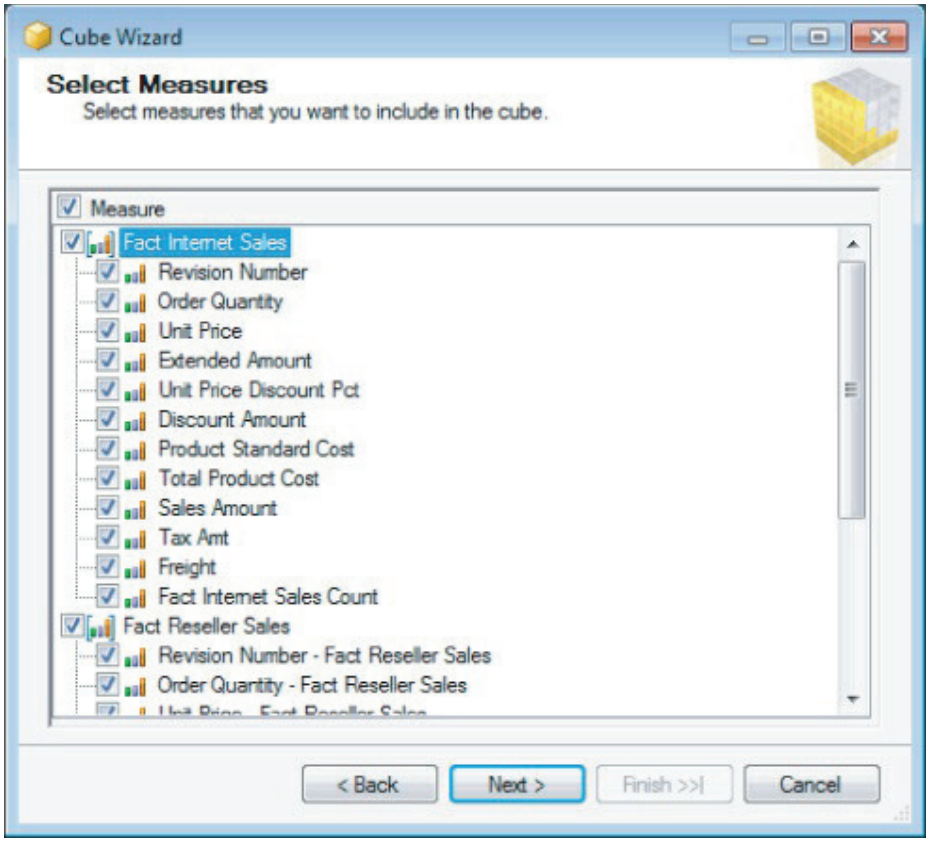
Figure 6.3



5. You can check or uncheck a table to identify it as a measure group table. The wizard suggests that the FactInternetSales, FactResellerSales, and DimReseller tables are measure group tables. The DimReseller table was detected as a measure group table because there is an outgoing relationship from it. However, it is not used as a measure group table in this example. Uncheck the DimReseller table and click Next.

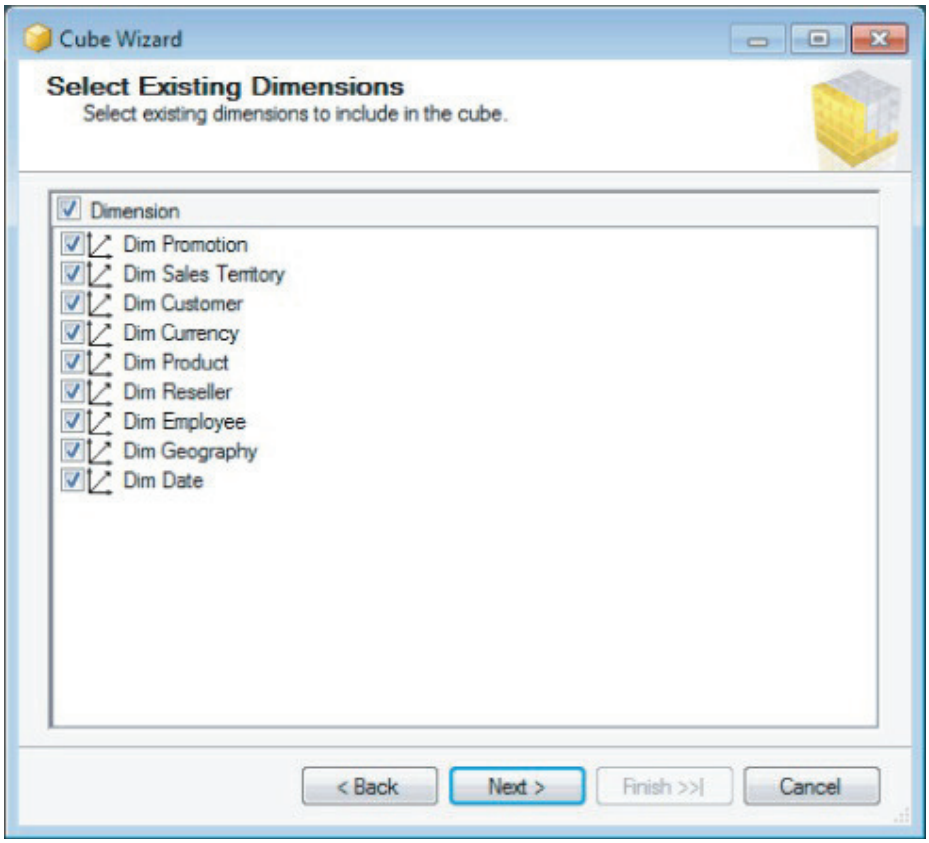
6. On the Select Measures page, the Cube Wizard shows all the columns from the fact tables that it detects as potential measures of the cube, as shown in Figure 6.4. The Cube Wizard does not select the primary and foreign key columns in a table as potential measures. There is a one-to-one mapping between a column in the fact table and a measure in the cube. There is one measure group for each fact table included in the cube. In the DSV you use, two fact tables exist, and therefore two measure groups — Fact Internet Sales and Fact Reseller Sales — are created. You can select or deselect the measures you want to be part of the cube in this page. Use the default selection on this page, and click Next.

Figure 6.4



7. In the Select Existing Dimensions page (Figure 6.5), the Cube Wizard displays a list of all dimensions defined in the project. Accept the selection of all the dimensions, and click Next.

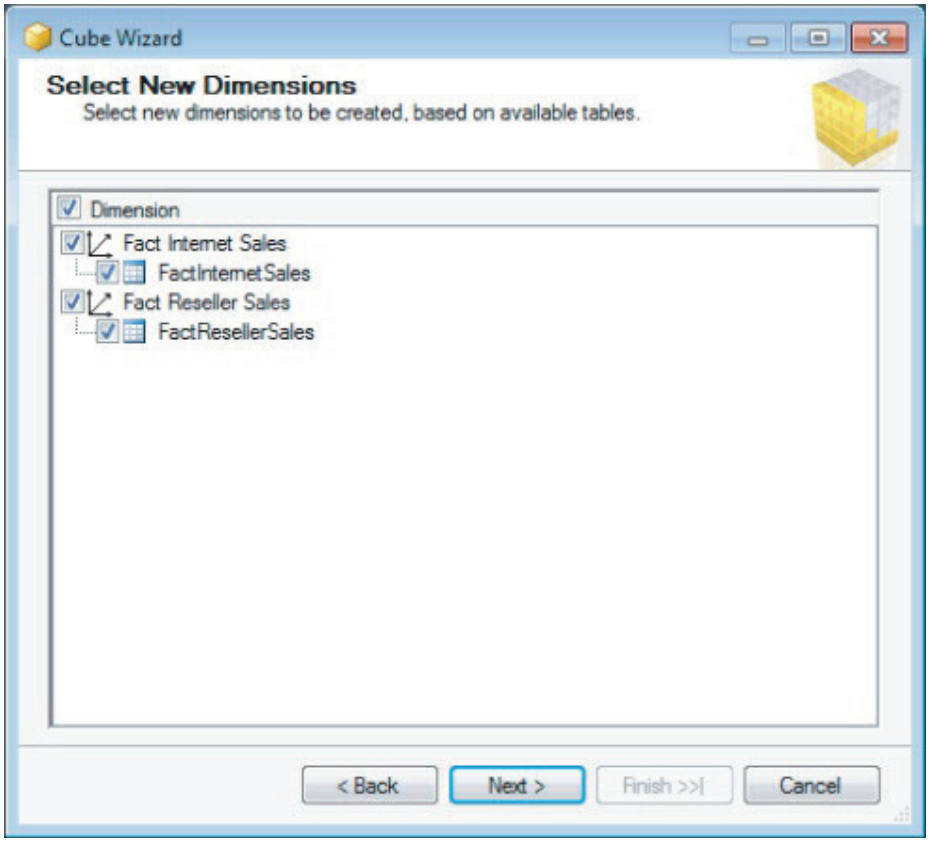
Figure 6.5



8. The Cube Wizard asks you to select any new dimensions to be created from existing tables in the data source that are not

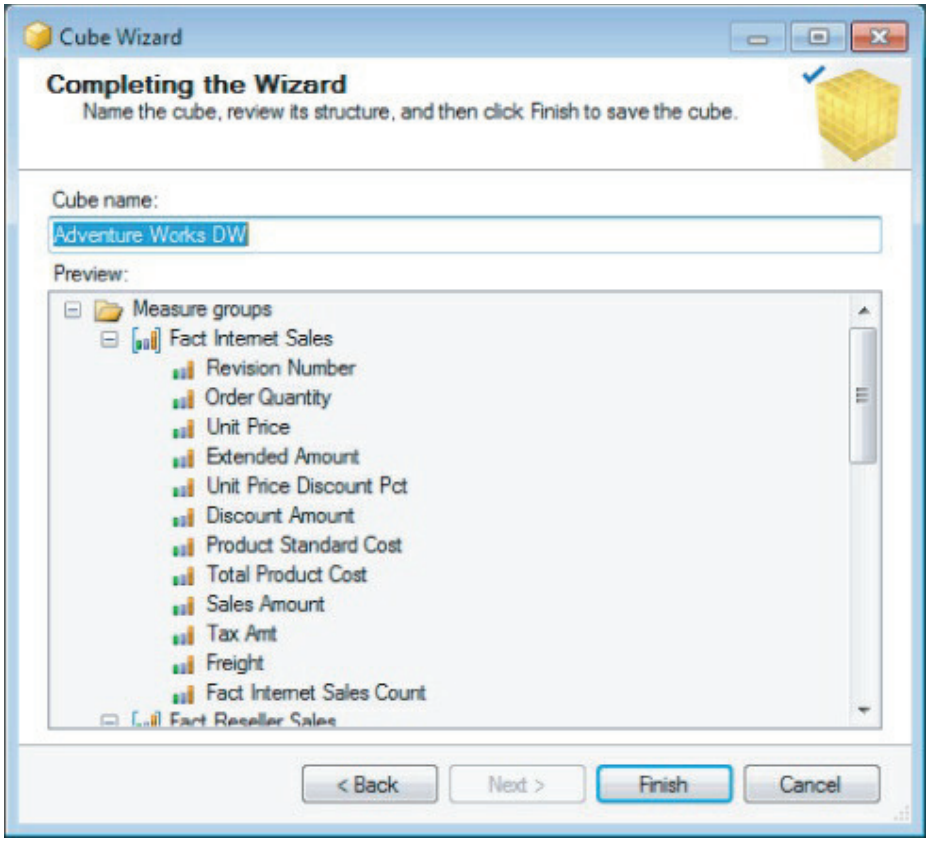
already used for dimensions, as shown in Figure 6.6. This illustration uses the fact tables only as measure groups and not for dimensions, so uncheck the Fact Reseller Sales and Fact Internet Sales dimensions on this page, and click Next.

Figure 6.6



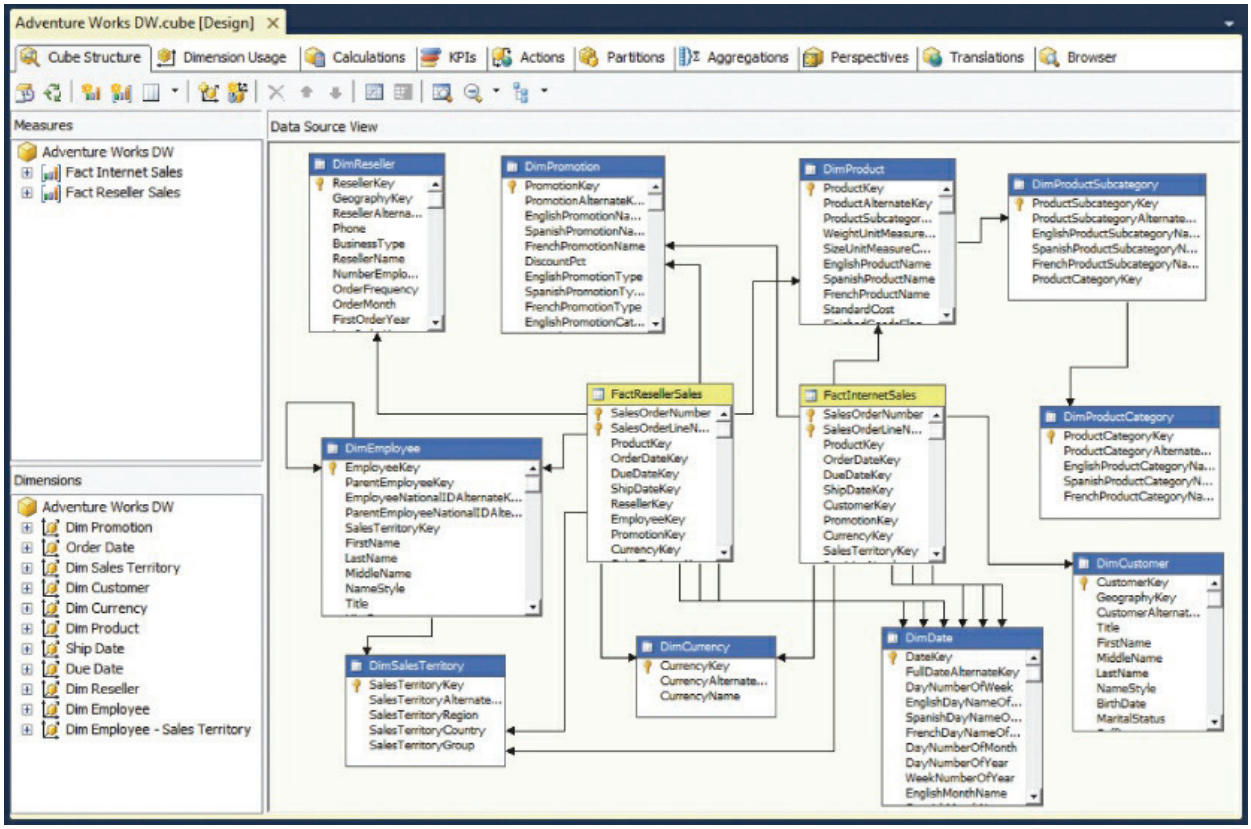
9. On the final page of the Cube Wizard (shown in Figure 6.7) you can specify the name of the cube to be created and review the measure groups, measures, dimensions, attributes, and hierarchies. Use the default name Adventure Works DW suggested by the Cube Wizard, and click Finish.

Figure 6.7



The Cube Wizard creates the cube when you click the Finish button. The created Adventure Works DW cube opens in the Cube Designer, as shown in Figure 6.8. The Cube Designer contains several pages that help perform specific operations that refine the initial cube created by the Cube Wizard. The default page is the Cube Structure page (refer to Figure 6.8). On the Cube Structure page, you can see three panes that show the Measures, Dimensions, and the Data Source View tables used in the cube. Operations such as adding or deleting tables in the DSV and zooming in or out with the DSV Designer are possible within the Cube Designer's Data Source View pane. The Dimensions pane shows the dimensions that are currently part of the cube, and the Measures pane shows the cube's measure groups and measures. You can add or delete measures and dimensions on the Cube Structure page. The dimensions in the cube are called cube dimensions. You can have multiple cube dimensions based on a single database dimension. For example, both the FactInternetSales and FactResellerSales fact tables have a relationship with the Dim Date dimension through their Order Date, Ship Date, and Due Date key fields. Hence you can see three cube dimensions: Ship Date, Due Date, and Order Date in the Dimensions pane, which refer to the Dim Date database dimension. A dimension such as Dim Date, which plays the role of multiple different cube dimensions, is called a role-playing dimension. You learn more about role-playing dimensions in Chapters 8 and 9. Within the Dimensions pane you can see the Hierarchies and Attributes of each dimension under separate folders when you expand each dimension.

Figure 6.8



So far you have created an Analysis Services database containing the Adventure Works DW cube. You must deploy the project to the Analysis Services instance before you can analyze the data within the cube. You can deploy the project to the server in one of the following ways:

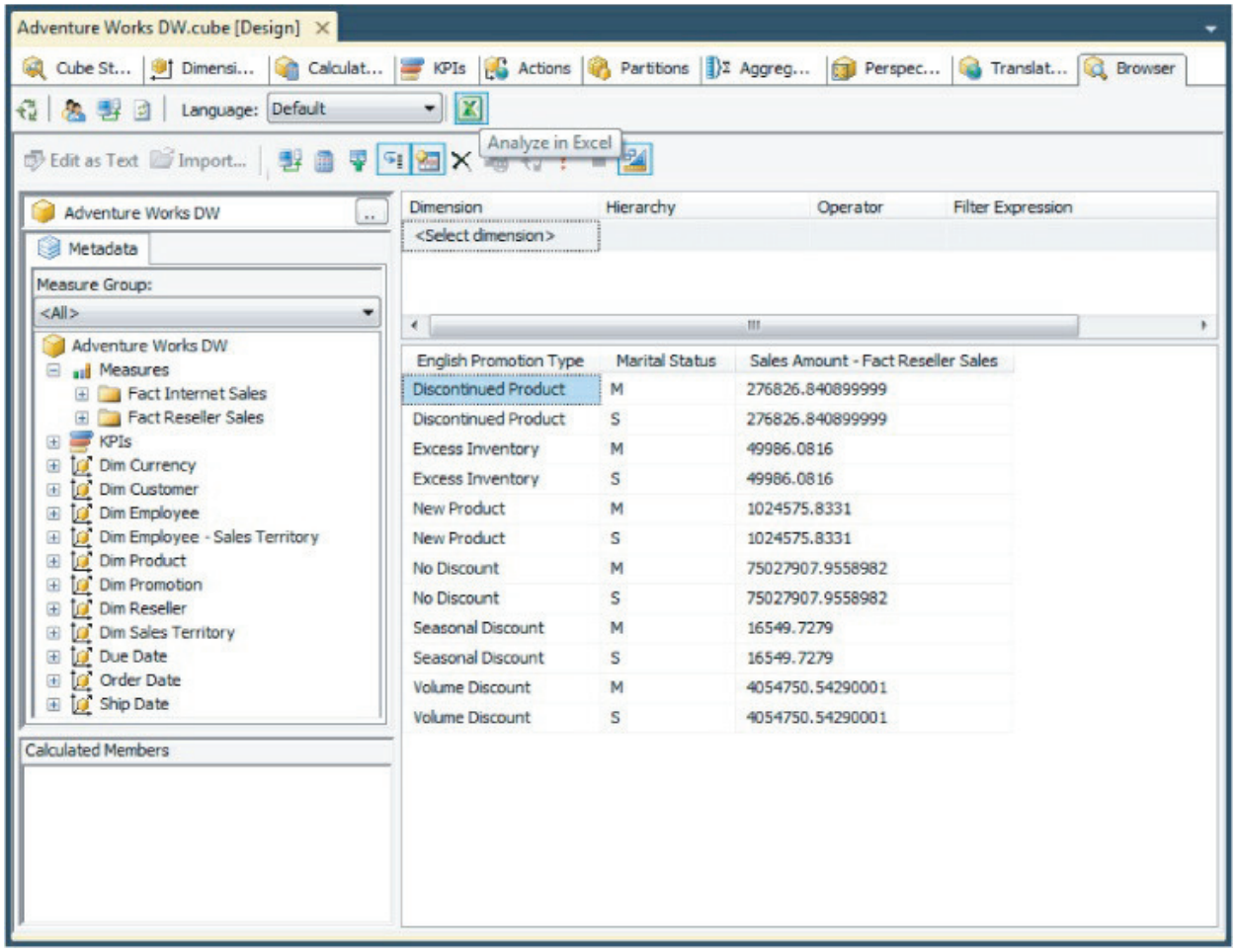
1. From the main menu, select Debug ⇒ Start Debugging.
2. In the Solution Explorer, right-click the AnalysisServicesMultidimensionalTutorial project node, and select Deploy.
3. Right-click the Adventure Works DW cube and choose Process — from which you will first be prompted to deploy the project, followed by the Process dialog to process the cube.
4. Use the shortcut key F5 to deploy and process.

When you deploy the project to the Analysis Services instance, SSDT sends an XMLA request containing object definitions to the Analysis Services server selected in the project. By default, the Analysis Services project deploys to the default instance of Analysis Services on your machine. The object definitions are of the cubes and dimensions you created. If you have installed Analysis Services as a named instance, you need to change the deployment server name. When the definitions of the objects in your project deploy to the server, SSDT sends another request to process the objects within the database.

Browsing Cubes

Now that you have deployed the cube to an Analysis Services instance, switch to the Cube Designer's Browser page. In the Browser page you can see panes on the left that enable you to select items for viewing and panes on the right that enable you to view and filter the results you chose on the left, as shown in Figure 6.9.

Figure 6.9



The Analysis Services 2012 Cube Browser

If you have used the cube browser in previous versions of Analysis Services, you will notice that the cube browser in Analysis Services 2012 is different than what you are used to seeing and working with. That is because a key component used in the implementation of previous versions of the cube browser, the Office Web Components, or OWC, is now discontinued and unsupported. It can no longer be shipped, either stand-alone or as a component of another product.

The Analysis Services team has, in place of the OWC, used the Analysis Services MDX query designer component (which is also used in Report Builder 3.0 and the PowerPivot Table Import Wizard) to implement cube browsing functionality in 2012 SSDT. Because the component was designed to be a query designer, it has some issues when used as a cube browser. For example, it enables you to create calculated members that aren't persisted after your browsing session is over. It also doesn't allow you to browse in the traditional row/column format of a PivotTable; you can only add columns to the tabular view.

The Analysis Services 2012 Cube Browser also includes a button, Analyze in Excel, which, if you have Excel installed on your machine, allows you to easily launch Excel with a PivotTable already connected to your deployed Analysis Services cube for testing and analysis.

When should you use the Cube Browser versus Analyze in Excel? The Cube Browser is more appropriate when you check certain features of your cube that are not easily shown in Excel. For example, if you want to check your cube's translations (which you learn about later in this chapter) in Excel you need to either be running on a localized version of the operating system that matches the translation you want to check or you need to manually modify the connection string Excel uses to talk to the Analysis Services server to include the appropriate `LocaleIdentifier` property and refresh the connection (for example, to show the French (France) translation you would modify the connection string to add `";LocaleIdentifier=1036"`). In the Cube Browser, it's simply a matter of selecting the wanted language in the Language drop-down on the toolbar.

On the other hand, Analyze in Excel is more appropriate when you want to test your cube using all the features of a powerful BI client such as Excel. Chances are that your end users will use Excel to work with your application when it is released.

On the upper left of the Cube Browser page, a control enables you to select the cube or perspective you want to browse. Below the cube selector is the Metadata pane, which enables you to select the measures, KPIs, and dimensions to browse in the Data pane. You can drag and drop measures and hierarchies from the Metadata pane onto the data area to analyze the data.

The right side of the Cube Browser is where you can view the data you want to analyze. There is also a Filter pane (above the data area) that you can use to filter the data being analyzed. The Filter pane enables you to use comparison operations such as equal, not equal, contains, in, not in, begins with, range operations, and any MDX expression to build a filter expression to help you analyze your multidimensional data.

In addition to the browsing functionality of the Cube Browser in SSDT, Analysis Services 2012 includes the capability of browsing the contents of your cube using Excel. You can accomplish this via the Analyze in Excel button on the Cube Browser's toolbar or the Analyze in Excel item in the Cube menu when you are in the Browser tab of the Cube Designer. For this feature to work correctly, you must have Excel installed on the machine you are running SSDT on. We recommend installing Excel to enable this very helpful feature.

Suppose you want to use the Analyze in Excel feature to analyze the Internet sales of products based on the promotions offered to customers and the marital status of those customers. First, you would click the Analyze in Excel toolbar button to bring up Excel with a PivotTable connected to your cube. Click Enable in the Security Notice dialog to enable data connections to the workbook. Excel opens with a blank PivotTable connected to your Analysis Services cube.

In the PivotTable Field List, expand the More fields folder under the Dim Promotion dimension, and drag and drop the English Promotion Type attribute to the Row Labels area in the PivotTable Field List. You learn the MDX statements that are generated by Excel in this section. The SQL Server Profiler has the capability to trace the MDX statements sent to Analysis Services instances. For more information on how to obtain traces, refer to the section on using SQL Server Profiler in Chapter 15.

Dropping the English Promotion Type attribute into the Row Labels section of the PivotTable Field List causes Excel to send the following MDX query to the Analysis Services instance:

```
SELECT NON EMPTY Hierarchize(
  {DrilldownLevel(
   {[Dim Promotion].[English Promotion Type].[All]},,,INCLUDE_CALC_MEMBERS
   )}
 ) DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME ON COLUMNS
FROM [Adventure Works DW] CELL PROPERTIES VALUE
```

Here you can see that Excel asks for the child members of the English Promotion Type attribute (via the `DrilldownLevel()` MDX function). It displays the results of that query in the Row Labels column of the PivotTable. At this point you cannot see any measure values in the Excel PivotTable, but if you run the above query in SSMS, some measure values will be returned. Those values will be those of the default measure for the cube (which is Order Quantity) but Excel doesn't presume that's a measure you want to see unless you perform some action to indicate it explicitly. Note the `ON COLUMNS` clause in the query even though Excel displays the results in rows. This is because `COLUMNS` is just a synonym for "the first axis." (Excel could have also said `ON 0`.)

Next, drag and drop the Marital Status attribute from the Dim Customer folder in the PivotTable Field List to the Excel Column Labels area. Excel now sends an expanded MDX query that includes two axes clauses, one for Marital Status and one for English Promotion Type.

```
SELECT NON EMPTY Hierarchize(
  {DrilldownLevel(
   {[Dim Customer].[Marital Status].[All]},,,INCLUDE_CALC_MEMBERS
   )}
 ) DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME ON COLUMNS,
 NON EMPTY Hierarchize(
  {DrilldownLevel(
   {[Dim Promotion].[English Promotion Type].[All]},,,INCLUDE_CALC_MEMBERS
   )}
 ) DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME ON ROWS
FROM [Adventure Works DW] CELL PROPERTIES VALUE
```

Excel now displays the members of both attributes in rows and columns.

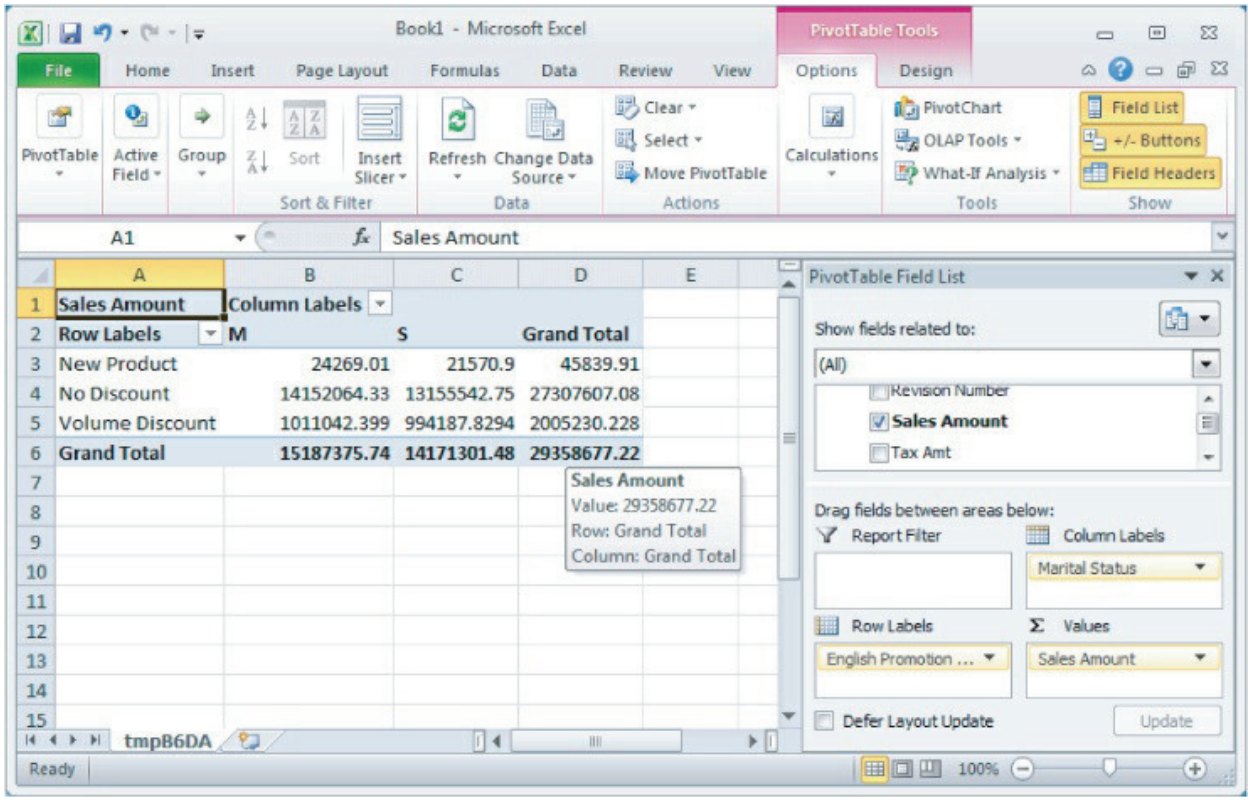
Finally, drag and drop the Sales Amount measure from the Fact Internet Sales measure group to the Values area of the PivotTable Field List. Excel sends the following query to your Analysis Services instance:

```
SELECT NON EMPTY Hierarchize(
  {DrilldownLevel(
   {[Dim Customer].[Marital Status].[All]},,,INCLUDE_CALC_MEMBERS
   )}
 ) DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME ON COLUMNS,
 NON EMPTY Hierarchize(
  {DrilldownLevel(
   {[Dim Promotion].[English Promotion Type].[All]},,,INCLUDE_CALC_MEMBERS
   )}
 ) DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME ON ROWS
FROM [Adventure Works DW]
WHERE ([Measures].[Sales Amount]) CELL PROPERTIES VALUE
```

Note the only difference between this query and the previous one is the addition of a `WHERE` clause specifying the measure you added to the Values field of the field list. Now Excel knows that you want a particular measure, so it asks for and displays it in the PivotTable.

If you hover over a particular cell in the PivotTable, you can see the cell values without formatting, along with the row and column member values that correspond to that cell, as shown in Figure 6.10.

Figure 6.10

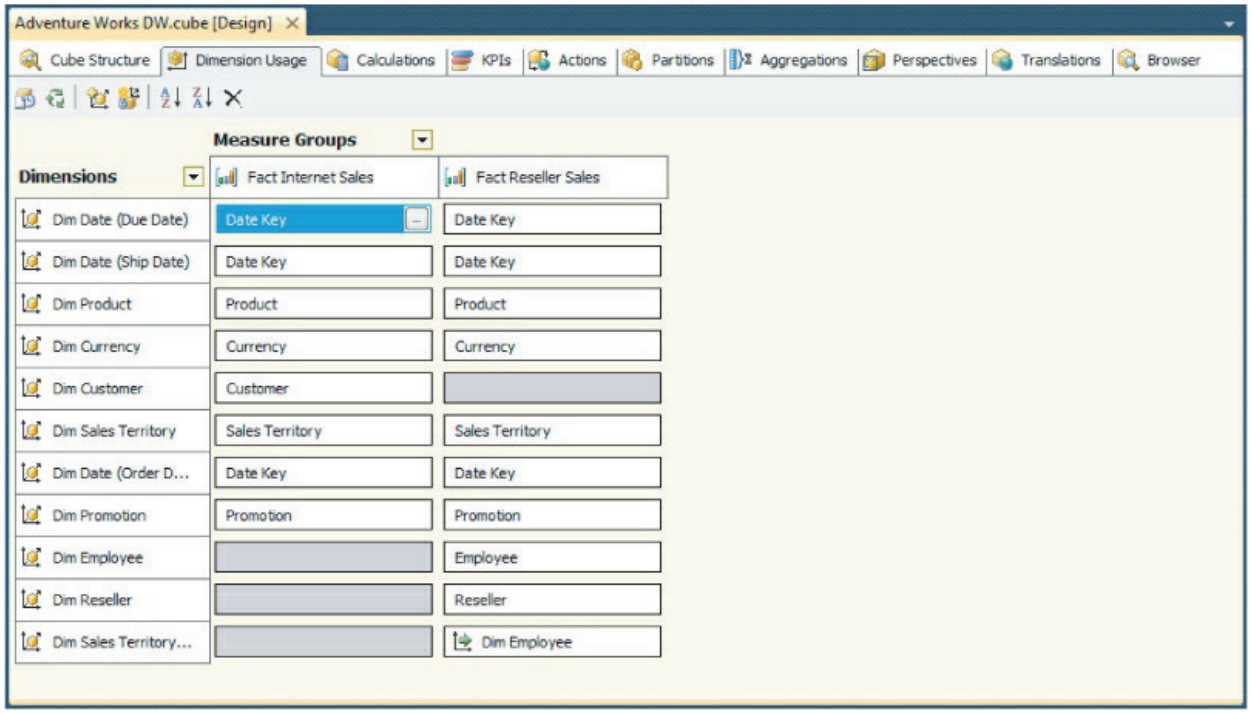


Cube Dimensions

The Cube Wizard helps you create your cube object from the DSV by creating appropriate dimension objects. The wizard detects the relationships between dimension tables and fact tables in the DSV, creates appropriate dimensions if needed, and establishes appropriate relationships between the dimensions and measure groups within the cube. The relationships between dimensions and measures groups define which dimensions in the cube can be used to slice and dice a measure group's data. As mentioned in the previous section, a cube contains instances of database dimensions referred to as cube dimensions. There can be multiple instances of a database dimension within a cube. There are relationships between a cube dimension and the measure groups within the cube. In this section you learn about various types of relationships between the cube dimensions and measure groups, as well as refine the Adventure Works DW cube created by the Cube Wizard by adding a new dimension.

The Cube Wizard establishes relationships between the measure groups and cube dimensions based on its analysis of relationships in the DSV. You might have to refine these relationships based on your business needs. You can change these relationships on the Dimension Usage tab of the cube editor. If you switch to the Dimension Usage tab, you see the Dimensions, Measure Groups of the cube, and the relationships between them, as shown in Figure 6.11.

Figure 6.11



The cube dimensions and measure groups are represented in a matrix format as rows and columns, respectively, where the relationship between them corresponds to the intersection cell. The intersection cell shows the dimension type along with the attribute used in the relationship to join.

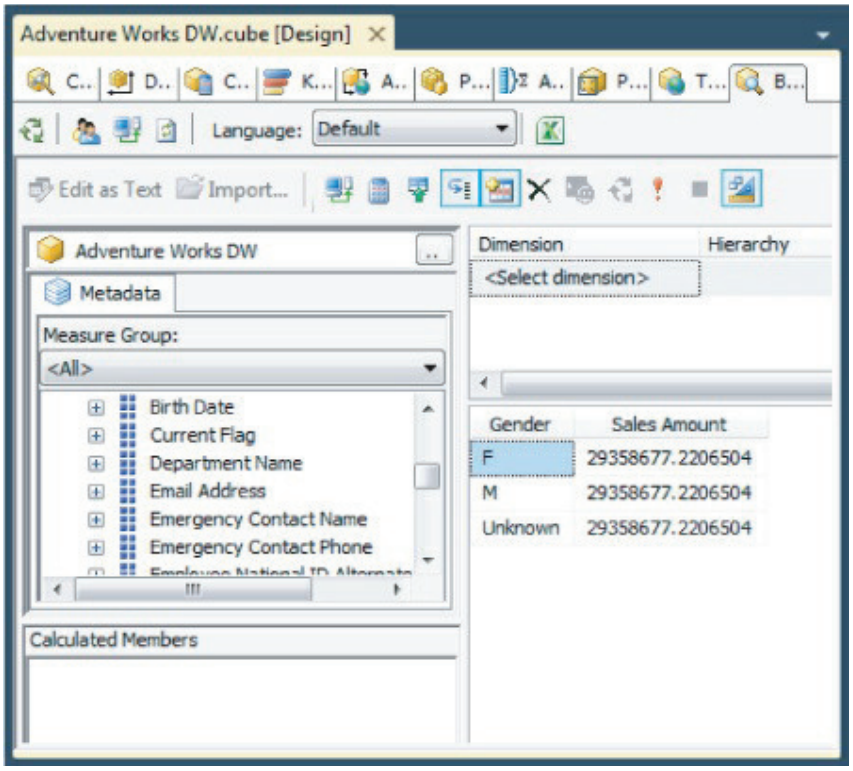
Relationship Types

Six different types of relationships can exist between a dimension and a measure group: No relationship, regular, fact, many-to-many, data mining, and referenced. In Figure 6.11 you see three of the six relationship types: No relationship (indicated by a gray box with no text), referenced (indicated by the glyph you see in the box at the intersection of the Fact Reseller Sales measure group and the Dim Sales Territory dimension), and regular (indicated by a white box with the name of the attribute used in the join that forms the relationship). The following sections describe each relationship type.

No Relationship

Cells shaded gray indicate no relationship exists between the dimension and measure group. Whenever no relationship exists between a dimension and measure group, the measure group property `IgnoreUnrelatedDimension` controls the results of queries involving any hierarchy of that dimension and any measure from the measure group. The measure values can either be null (`IgnoreUnrelatedDimension=False`) or the same value for each member of the dimension (`IgnoreUnrelatedDimension=True`). For example, no relationship exists between the Dim Employee dimension and the Fact Internet Sales measure group. If you browse the Gender hierarchy of Dim Employee and the Sales Amount measure, you see that the measure values for each member of Gender hierarchy are the same value as the Grand Total, as shown in Figure 6.12. This is because the `IgnoreUnrelatedDimension` property is set to `True` by the Cube Wizard as a default. You learn more about properties of measure groups and measures later in this chapter.

Figure 6.12



Regular Relationships

Cells corresponding to a specific dimension and measure group can have an attribute specified that indicates that the dimension type is Regular. Further, such attributes can be used in the join condition between the dimension and the measure group. Often this attribute is the key attribute of the dimension and is called the granularity attribute. The granularity attribute can be an attribute at a higher level of granularity than the key attribute of the dimension. When you browse a dimension along with measures of a measure group where the dimension and measure group have a regular relationship, Analysis Services aggregates the data appropriately. If you have your granularity attribute above the key attribute of your dimension, it is critical that you define appropriate attribute relationships in your dimension to make sure the data getting aggregated is accurate. The relationship between Dim Customer and Fact Internet Sales measure group is a regular relationship. The granularity attribute is shown in the cell intersecting the dimension and measure group, as shown in Figure 6.11.

Fact Relationships

When a table is used as both a fact and dimension table, a unique relationship exists between the dimension and measure group called a fact relationship. The relationship is similar to that of the regular dimension, but specifying it as a fact dimension helps improve query performance for a certain class of MDX queries. You learn more about fact dimensions in Chapter 9.

Many-to-Many Relationships

Typically, a one-to-many relationship exists between a dimension and a fact member for regular relationships. When you have a one-to-one relationship between a fact and a dimension member, you typically have a fact relationship. When a many-to-many relationship exists between a fact and a dimension member, the dimension member has a one-to-many relationship with various facts, and a single fact is associated with multiple dimension members. The definition for a many-to-many relationship can be understood via an example: Assume you have a fact table for sales of books that is related to a dimension table containing author information. Authors can have multiple books and books can have multiple authors. In order to model that many-to-many relationship you need to have both an intermediate fact table related to the book sales and authors and an intermediate dimension table related to both fact tables. You learn about how to work with many-to-many relationships in Chapter 9.

Data Mining Relationships

Data mining dimensions are another item type in the list of relationships; these are used to establish linkage between a cube and a dimension created from a data mining model. You learn more about this in Chapters 9 and 12.

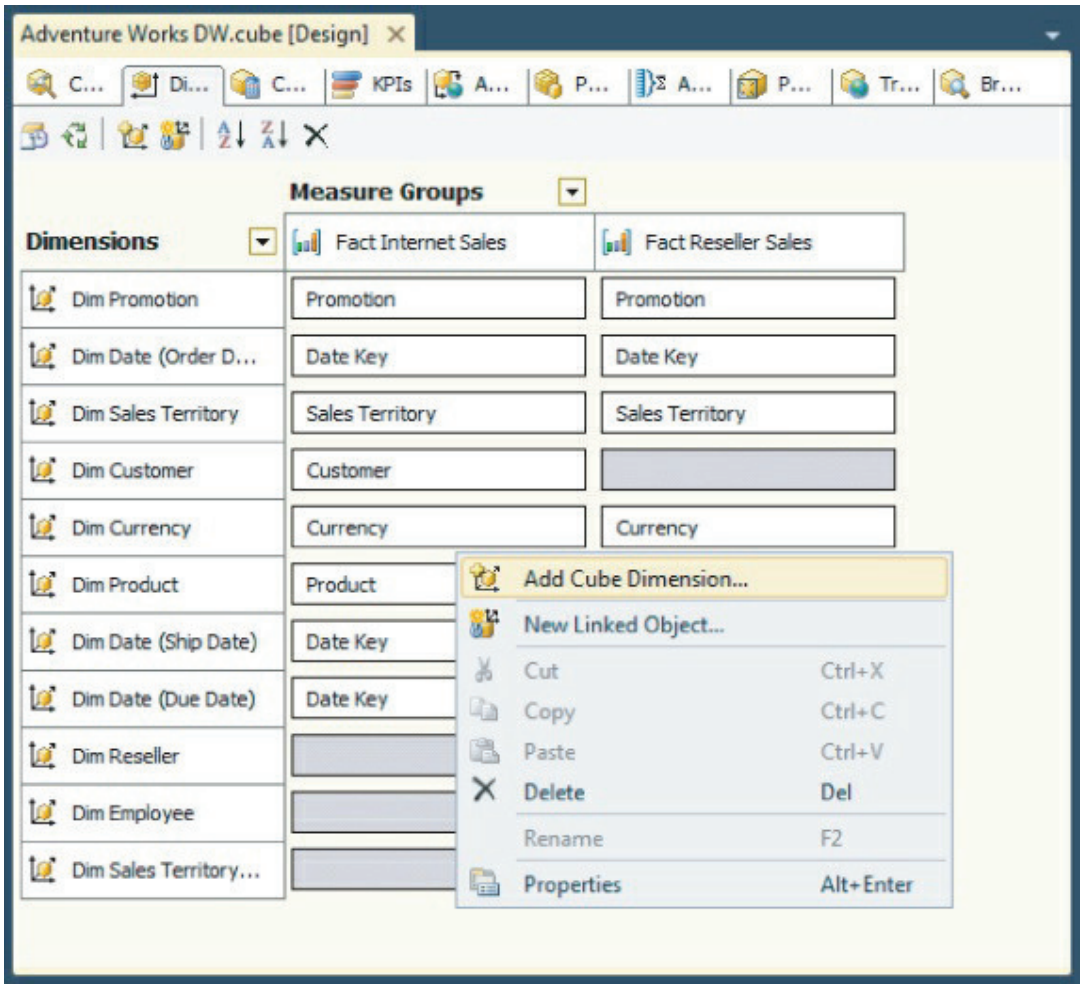
Referenced Relationships

When a dimension is related to the fact data through another dimension, you define this indirect relationship between the measure group and the dimension as a referenced relationship. In Figure 6.11 the Dim Sales Territory dimension is related to the Fact Reseller Sales measure group through the Employee dimension. The icon at the intersection of the dimension and measure group indicates this referenced relationship. You might also recall that you added the Dim Geography dimension in the Select Existing Dimensions page of the Cube Wizard. However, the Cube Wizard was not smart enough to figure out there is a relationship between the fact tables and the Dim Geography dimension table through other dimension tables. Hence the Cube Wizard did not add the Dim Geography dimension as a cube dimension. Because the relationship between the Dim Geography dimension and the measure groups in the Adventure Works DW cube is through another dimension, you can say that there is an indirect relationship between the Dim Geography dimension and the measure groups.

Follow these steps to add the Dim Geography dimension to the cube and establish the referenced relationship:

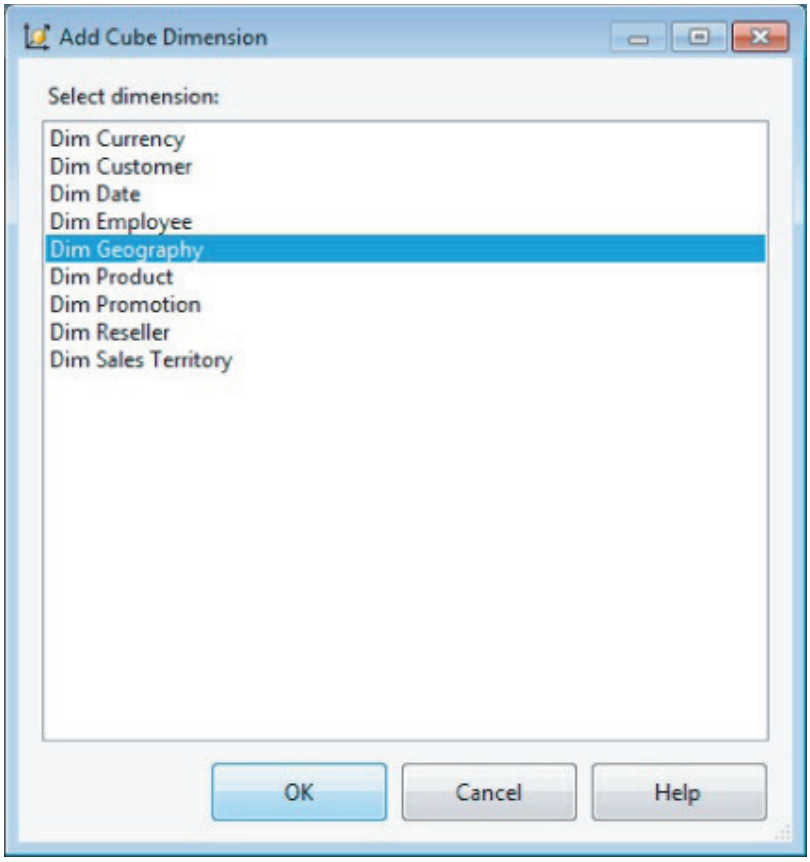
1. To add the Dim Geography database dimension to the cube, right-click anywhere in the Dimension Usage page, and select Add Cube Dimension, as shown in Figure 6.13.

Figure 6.13



2. A dialog showing all the dimensions within the project launches, as shown in Figure 6.14. Select the Dim Geography dimension, and click OK.

Figure 6.14

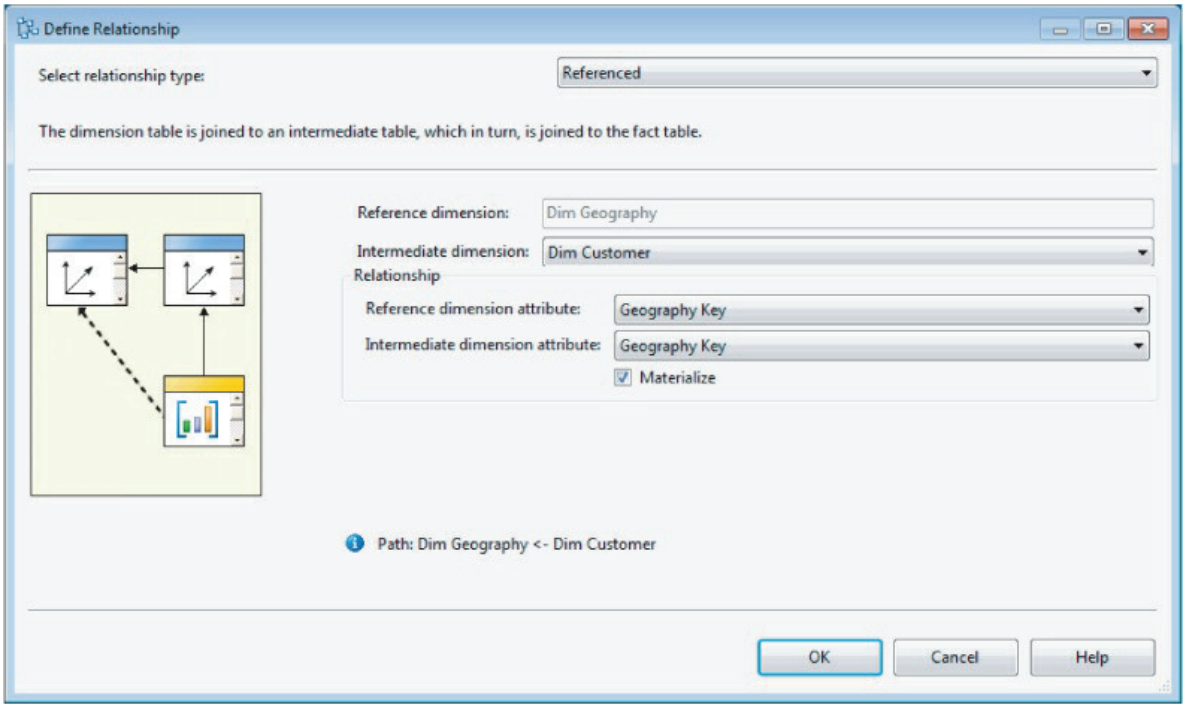


The Cube Designer cannot identify a relationship through an attribute between the existing measure groups and the Geography dimension and as a result leaves the relationship definition up to you. There exists an indirect relationship between the Dim Geography dimension and Fact Internet Sales measure group through the Dim Customer dimension. There is an indirect relationship between the Dim Geography dimension and the Fact Reseller measure group through the Dim Reseller dimension. You need to define these referenced relationships.

3. To define the relationship between the Dim Geography dimension and the Fact Internet Sales measure group, select the corresponding cell in the matrix, and you see a button with an ellipsison the right side of that cell. Click the Ellipsis button (…).

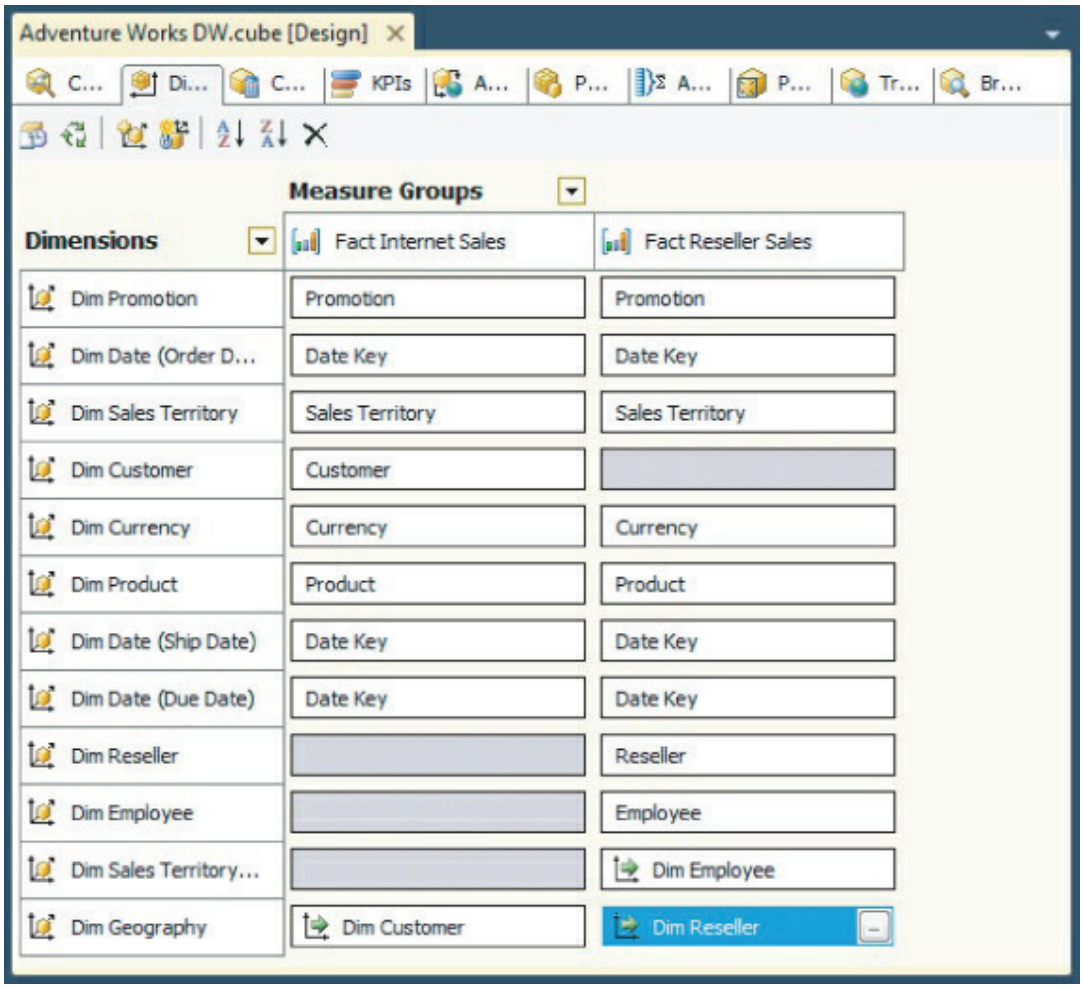
4. This opens the Define Relationship dialog shown in Figure 6.15. On the Select relationship type drop-down, select Referenced. The Dim Geography dimension forms an indirect or referenced relationship with the Fact Internet Sales measure group through the Dim Customer dimension. You define the intermediate dimension through the Intermediate dimension field in the dialog. After you define the intermediate dimension, you need to select the attributes involved in the join of the relationship. The Reference dimension attribute is the attribute in the reference dimension that is used in the join between the intermediate dimension (Dim Customer) and the reference dimension (Dim Geography). The Intermediate dimension attribute is the attribute of the intermediate dimension that is involved in the join between the reference dimension and the intermediate dimension. Select the Intermediate dimension as Dim Customer, the Reference dimension attribute as Geography Key, and the Intermediate dimension attribute as Geography Key as shown in Figure 6.15, and click OK. In Figure 6.15 you see a check box with the text Materialize. This check box is enabled by default in SQL Server Analysis Services. By enabling this check box, you ensure that Analysis Services builds appropriate indexes to get improved query performance when querying fact data along with reference dimension hierarchies.

Figure 6.15



5. Similar to step 4, establish a referenced relationship between the Dim Geography dimension and the Fact Reseller Sales measure group through the Dim Reseller dimension. After you specify the relationship between Dim Geography and the two measure groups of the cube, your Dimension Usage tab should resemble Figure 6.16.

Figure 6.16

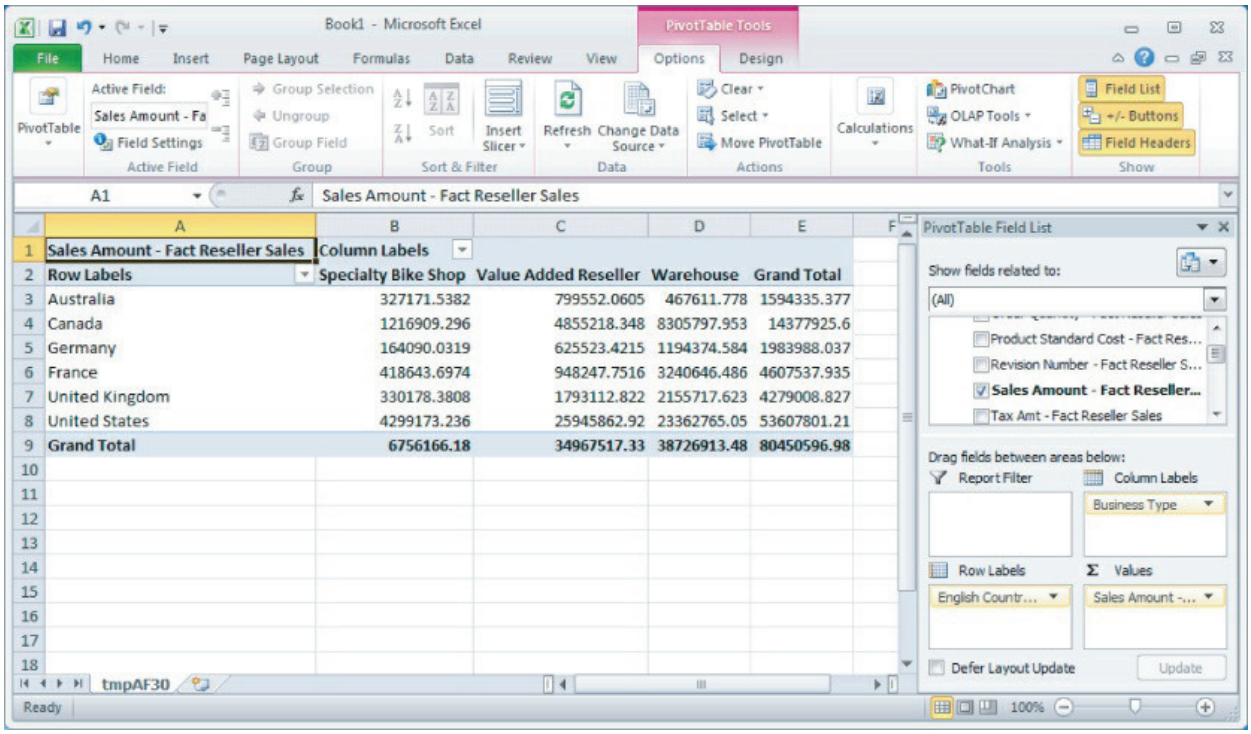


A referenced relationship between a dimension and a measure group is indicated by an arrow pointing to the intermediate dimension (refer to Figure 6.16). This graphical view of the reference relationship helps you identify the type of relationship between a dimension and measure group when you look at the Dimension Usage tab of the cube editor. Similar graphical representations are available for fact, many-to-many, and data mining dimensions, and you learn about these relationships in Chapters 9 and 12.

Browsing Reference Dimensions in Excel

Having added the Dim Geography dimension as a reference dimension to the cube, assume you want to analyze the Reseller Sales based on different business types in various countries. To do so you need to redeploy and process the cube with the changes. Then go to the Cube Browser, and click the Analyze in Excel button. In the Excel PivotTable field list, expand the More Fields folder under the Dim Geography dimension, and drag and drop the English Country Region Name attribute to the Row Labels area in the PivotTable Field List. Expand the More Fields folder under the Dim Reseller dimension, and drag and drop the Business Type attribute to the Column Labels area. Finally, drag and drop the Sales Amount – Fact Reseller Sales measure of the Fact Reseller Sales measure group to the Values area. You can now analyze the Sales data based on the business type in each country, as shown in Figure 6.17. Based on this sales knowledge — the costs associated with the products and your business goals — you can strategically promote the business type yielding the maximum profit for your company. Reference dimensions help you to analyze fact data even though they are not directly related to the facts.

Figure 6.17



Excel sends the following query to retrieve data for analyzing the reseller sales fact of various business types across various countries of the resellers.

```
SELECT
 NON EMPTY Hierarchize(
  {DrilldownLevel({[Dim Reseller].[Business Type].[All]}
   ,,,INCLUDE_CALC_MEMBERS)})
    DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME ON COLUMNS ,
 NON EMPTY Hierarchize(
  {DrilldownLevel({[Dim Geography].[English Country Region Name].[All]}
   ,,,INCLUDE_CALC_MEMBERS)})
    DIMENSION PROPERTIES PARENT_UNIQUE_NAME,HIERARCHY_UNIQUE_NAME,
  [Dim Geography].[English Country Region Name].
   [English Country Region Name].[French Country Region Name],
  [Dim Geography].[English Country Region Name].
   [English Country Region Name].[Spanish Country Region Name] ON ROWS
FROM [Adventure Works DW]
WHERE ([Measures].[Sales Amount - Fact Reseller Sales])
 CELL PROPERTIES VALUE, FORMAT_STRING, LANGUAGE, BACK_COLOR,
  FORE_COLOR, FONT_FLAGS
```

Because the query used by Excel retrieves data on three-dimensional axes, you cannot execute the same query in SQL Server Management Studio (SSMS) because SSMS can display only two-dimensional results. Therefore, if you need to see the same results in SSMS, you need to use an MDX query that can retrieve results in a two-dimensional format. You can rewrite the MDX query generated by Excel using the `CrossJoin` function or the cross join operator (`*`) so that the results can be retrieved on two axes. A simplified MDX query that returns the same results as the Excel query follows:

```
SELECT { [Measures].[Sales Amount - Fact Reseller Sales]} ON COLUMNS,
 NON EMPTY { [Dim Geography].[English Country Region Name].MEMBERS *
 [Dim Reseller].[Business Type].MEMBERS} ON ROWS
FROM [Adventure Works DW]
```

This simplified query asks for the members of the dimension attributes rather than drilling down to them from the All member as Excel's query does.

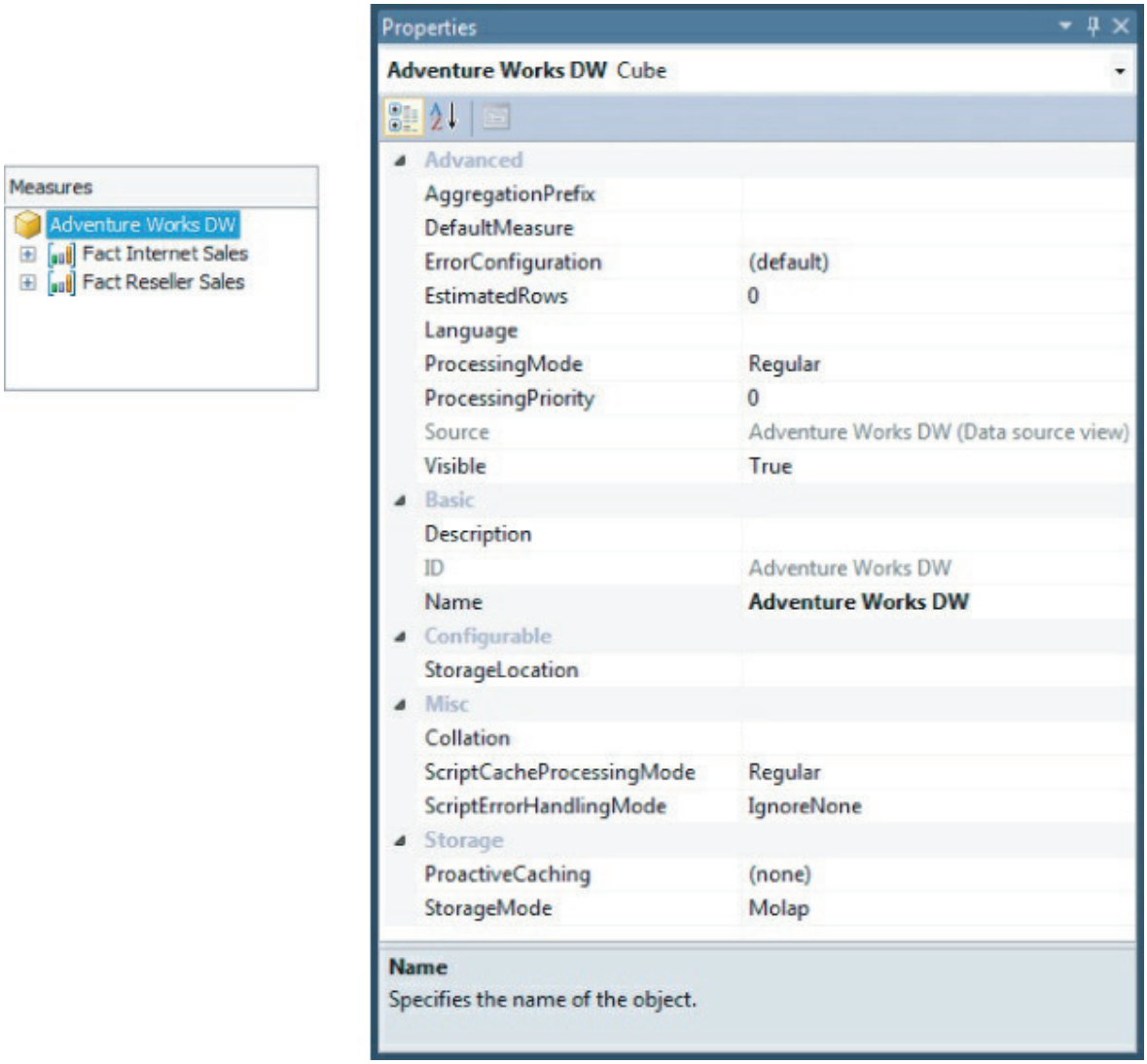
So far you have learned about cube dimensions, how to add them to a cube, how to define relationships between dimensions and measure groups, and then how to query data along with the dimensions. Cube dimensions and their attributes and hierarchies contain several properties. Some properties such as AttributeHierarchyEnabled, AttributeHierarchyVisible, and AttributeHierarchyOptimizedState reflect the default state of the cube dimension hierarchies or attributes. You can override these properties so that appropriate settings are applied for the dimensions within the cube. The AggregationUsage attribute property and AllMemberAggregationUsage dimension property control the behavior of aggregations designed on the cube. You learn more about these properties in Chapters 9 and 10.

Measures and Measure Groups

You learned about editing cube dimensions and establishing the right relationships between dimensions and measure groups. Similarly, you can add or delete a cube's measures and measure groups. Measures are the focus point for data analysis and therefore they are the core objects of a cube. Measures are columns from the fact table that contain meaningful information for data analysis. Usually, measures are numeric types that can be aggregated or summarized along the attributes and hierarchies of a dimension. You can specify the type of aggregation to be applied for each measure. The most widely used aggregate functions are `Sum`, `Count`, and `Distinct Count`. A collection of measures forms an object called a measure group, and a collection of measure groups forms the dimension called *Measures* in the cube. Measures is a keyword in Analysis Services that refers to a special dimension that contains only the fact data.

If you click the Cube Structure tab in the cube editor, you can see the Measures pane on the top-left corner. Select the cube named Adventure Works DW in the Measures pane to see the cube's properties in the Properties window located on the bottom-right corner of SSDT. Figure 6.18 shows the Measures and Properties panes. The Measures pane shows the cube name and the measure groups within the cube. You can see the two measure groups, Fact Reseller Sales and Fact Internet Sales, which correspond to the two fact tables. There is typically a one-to-one relationship between a fact table and measure group in the cube.

Figure 6.18



In your source data, if you had partitioned your fact data into multiple fact tables across a specific dimension, it needs to be handled differently when designing the cube. For example, if you have Fact Internet Sales data stored in separate fact tables for each quarter (fact data has been partitioned into multiple fact tables across the Time dimension), then, with respect to the cube, all these tables can be considered a single fact table because they have the same schema. You typically partition your relational fact data into multiple fact tables due to design or scalability considerations, but when you want to analyze the data, you aggregate the data appropriately across various dimensions, especially the Time dimension. You can either merge the data from all the fact tables within the DSV with a named query, or you can utilize the partitioning feature in Analysis Services so that Analysis Services aggregates the data correctly during browsing. You learn more about partitions in Chapters 7 and 10.

You can see several properties of the cube in Figure 6.18. An important property is DefaultMeasure. The reason why the default measure is important is that whenever your MDX query does not explicitly contain a member from the measures dimension, the default measure is returned. In addition, the default measure is used whenever restrictions are applied in the query with the `WHERE` clause. Based on the setting of the default measure property your results can be different. If you select the DefaultMeasure property, you can see a drop-down list that shows all the measures of the cube. You can choose the measure you want to define as the default measure of the cube from this list. If the default measure is not specified, the first measure of the first measure group of the cube (as seen in the Measures pane) will be used as the default measure of the cube.

Another important property is the StorageMode property. This defines whether your fact data will be stored in Analysis Services, your relational data source, or both. The StorageMode property has three options: Multidimensional OLAP (MOLAP), Relational OLAP (ROLAP), and Hybrid OLAP (HOLAP). The default value is MOLAP, which means that when the cube is processed, Analysis Services reads the relational data and stores it in the Analysis Services database for fast retrieval. You learn more about the defining storage modes in Chapter 9. You have the option to instruct the Analysis Services instance to automatically update cube and dimension objects if there were a change in the relational data. The ProactiveCaching property lets you specify the frequency of the update of the cube data based on changes in the relational source data.

The ErrorConfiguration property helps in handling the various errors that can occur while processing the fact data and defining what actions should be taken under such error circumstances such as ignoring the error, converting to a specific value, or stopping processing

when errors are encountered.

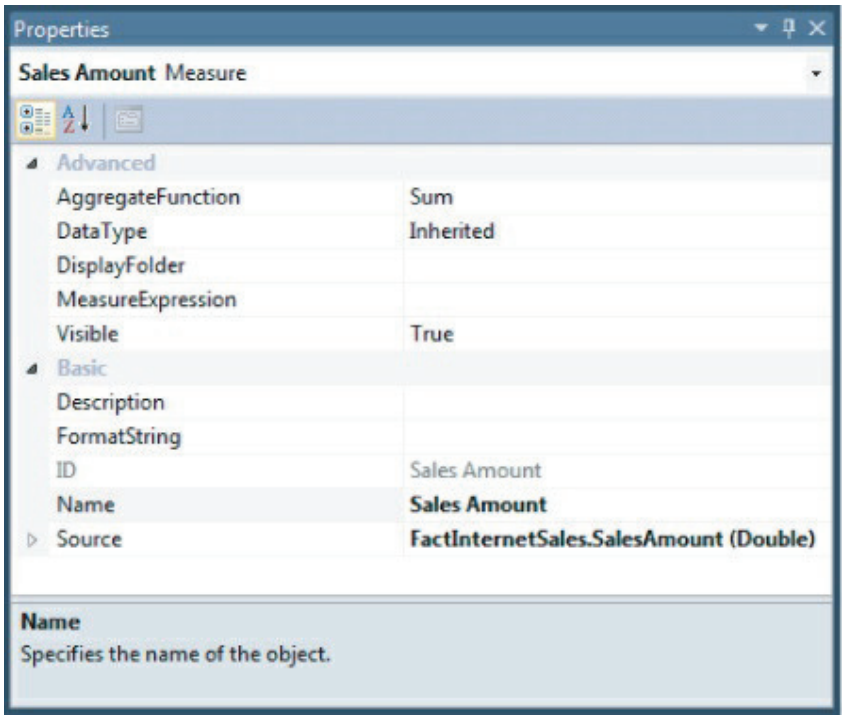
One of the main features of an OLAP database is the capability to create aggregations that facilitate fast query response times. You can use the AggregationPrefix property to prefix the name of the aggregations created for the cube.

The remaining properties are self-explanatory, and you can find detailed information for each property in Analysis Services 2012 product documentation.

If you select one of the measure groups in the Measures pane, you see the properties associated with that measure group in the Properties window. Most of the properties at the cube level are also applicable to the measure group level. If you specify a value for a common property at the measure group level that is different than the value for the same property at the cube level, the value specified at the measure group level will be applied to that measure group.

Expand the Fact Internet Sales measure group, and select the Sales Amount measure. The Properties window now shows the properties of the measure, as shown in Figure 6.19. Next, you learn the important properties of a measure in detail.

Figure 6.19



The AggregateFunction property defines how the measure value is to be aggregated from one level to another level of a hierarchy in a dimension. For example, assume a Product dimension contains a hierarchy called Products that contains two levels, Model Name and Product Name. Each model contains one or more products. If you want the sales amount of a specific product to be added to that of the parent model, you need to specify the `Sum` as the aggregate function. Whenever you browse the cube along the Products hierarchy, you can see that the sales of each product are added to the sales amount value of the corresponding model. However, sometimes you might not want the measure value to be summed while browsing a hierarchy. Analysis Services supports several different aggregate functions. Aggregation functions supported in the measure properties can also be implemented in MDX scripts. (You learn about MDX scripts in Chapter 9.) However, you should use the built-in aggregation functions shown in the Properties window to get optimal performance from your Analysis Services instance.

Other than the `Sum` aggregate function, the most commonly used aggregate functions are `Count` and `Distinct Count`. Use the `Count` aggregate function, as the name indicates, whenever you want to count each occurrence of the measure value rather than add the measure values. For example, if you want to find the number of transactions in a day or number of customers in a day, you would use a `Count` aggregate function on a fact table column that indicates the customers who came to the store on a specific day. The `Distinct Count` aggregate function, on the other hand, identifies the unique number of occurrences of a specific measure value. For example, a customer can buy a specific product every month. If you want to find the number of customers who purchase a specific product, use the `Distinct Count` aggregate function. You see examples of `Count` and `Distinct Count` aggregate functions later in this section. The `None` aggregate function is used when you do not want to aggregate the values of a specific measure across a dimension. An example of where the `None` aggregate function would be used is for the price of a specific product or discount provided on a unit product.

When you build and browse a cube, you see all the measures occurring under the dimension called Measures in the Metadata pane of the Cube Browser. If you want to organize the related measures in a logical structure that is more meaningful for users, use the DisplayFolder property. You can specify a measure to be part of one or more display folders by editing its DisplayFolder property. If you enter a name in the DisplayFolder property, that specific measure becomes part of a display folder with that name. You can make a specific measure part of multiple display folders by specifying the display folders' names separated by semicolons. When display folders are specified, while browsing the cube you can see the display folders under the appropriate measure group name in the Metadata pane

of the Browser. Therefore you cannot have measures from different measure groups in a single display folder.

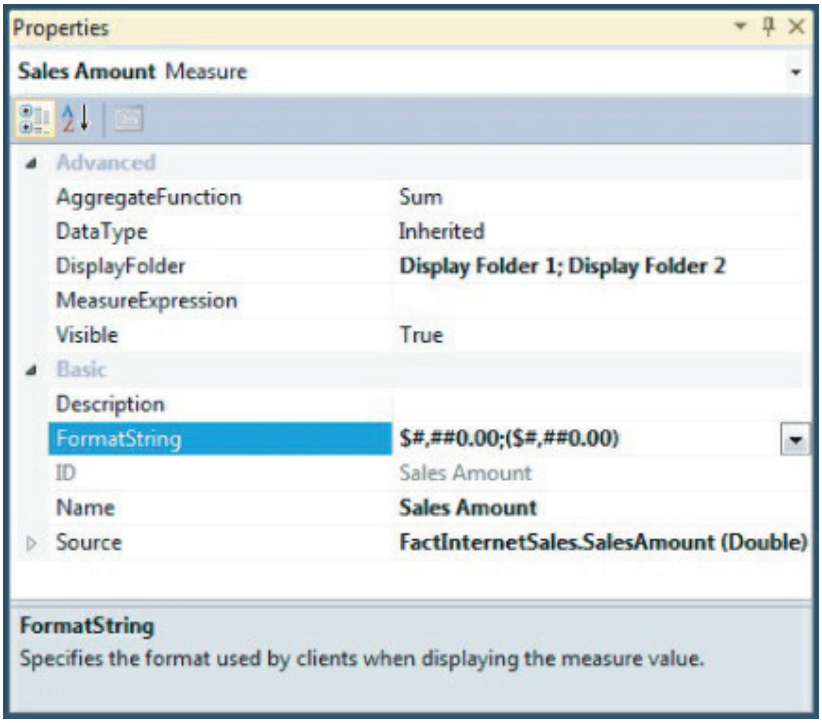
In some business applications you may want to allow access to only the aggregated results of a measure. For such applications you need a way to aggregate the results of a measure but not show the base measure. You can aggregate the results of a measure by specifying a measure called a calculated measure (you learn more about calculations later in this chapter) and hide the base measure. The Visible measure property allows you to hide the base measure from viewing for such applications.

The FormatString property allows you to show the measure value in a format of your choice. If you select the FormatString property you can see the various format options available.

The MeasureExpression property specifies the expression that evaluates the value for the measure. For example, if you have Sales information you might want to ensure the Sales information is presented in the local currency based on the currency conversion rates. In such a case you can define an appropriate expression for the MeasureExpression property for the measure Sales. You learn more about the MeasureExpression property in Chapter 9.

The easiest way to see the effect of the properties mentioned is to try them in your project. Specify two display folders named DisplayFolder 1 and DisplayFolder 2 for the Sales Amount measure in the Fact Internet Sales measure group. Because the Sales Amount measure is a currency data type, you can select the currency format. Select the $#,##0.00;($#,##0.00) format from the FormatString property drop-down. The Properties window for the Sales Amount measure should resemble Figure 6.20.

Figure 6.20

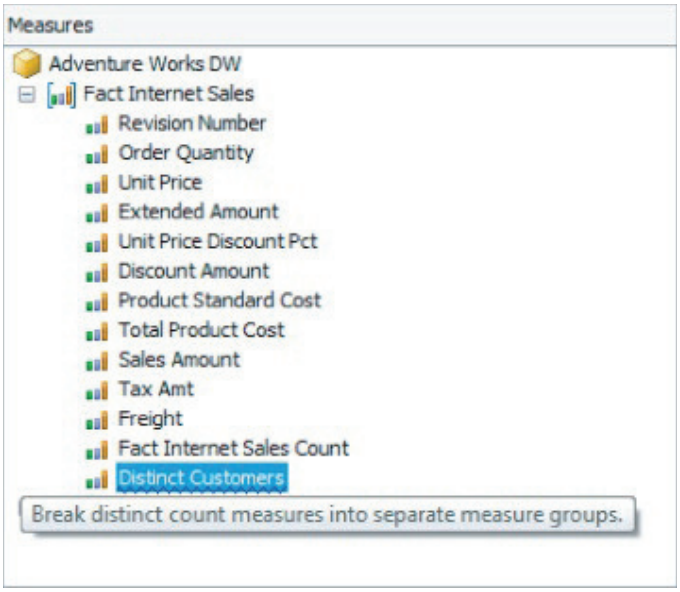


You learned examples of where the `Count` and `Distinct Count` aggregate functions can be useful. In your Adventure Works DW cube, if you want to count the number of customers and distinct customers who have bought specific products, you can use these aggregate functions. Customer Key identifies the customer who has bought a specific product in the fact table. Therefore, to see the customer counts mentioned, you need to create two new measures using the Customer Key fact table column. To create the two new measures follow these steps:

1. Right-click the Fact Internet Sales measure group, and select New Measure.
2. In the New Measure dialog, check the Show All Columns check box.
3. In the Source columns list, select Customer Key, and click OK. A new measure called Customer Key is created.
4. In the Measures pane, change the name for this measure from Customer Key to Distinct Customers by right-clicking the measure and selecting Rename.
5. Change the aggregate function for this measure to `Distinct Count`.

At this point you notice that a blue squiggly line shows up under the Distinct Customers measure. Analysis Services starting with the 2008 version has built-in checks for many of the best practices for dimension and cube design. These best practices are implemented as warnings in the Analysis Management Objects (AMO) API. SSDT surfaces these AMO warnings through warning icons and blue squiggly lines for objects that don't meet these design best practices. To discover the rules that generate these warnings, simply move the mouse over the blue squiggly line or warning icon to see a tooltip with the best practice design rule. In this case the rule suggests that distinct count measures should be broken out into separate measure groups (see Figure 6.21).

Figure 6.21



The reason for this warning is that distinct count measures are semi-additive and require storing records at a finer level of detail than measures using other aggregate functions. For a distinct count of customers to be accessible, the individual customer IDs must be available. Therefore, Analysis Services must use the Customer Key attribute for any and all aggregations. Consider the following example: If you are interested in the total Sales (sum) for a product in a given year, you do not need to retain the individual customer IDs that purchased the product within the chosen year. The Analysis Services engine can pre-aggregate data at the Product ID and Year attribute levels as part of the normal processing of the cube. However, if there is a requirement that you know the distinct number of customers who purchased the product within the year, you must retain the customer keys throughout any aggregated data.

If the product in question were purchased by 50,000 customers over the year, the aggregate goes from one row of data per product per year to 50,000 rows per product for the year. Separating `Distinct Count` measures into different measure groups allows maximum pre-aggregation of other additive measures with minimal storage and on-the-fly aggregation requirements.

For this illustration you can ignore this best practice warning and move on to the next steps.

1. Right-click the Fact Internet Sales measure group, and select New Measure.

2. In the New Measure dialog, select the Show All Columns check box.

3. Select the Customer Key column, and click OK. A new measure called Customer Key is now created.

4. In the Measures pane, change the name of this measure from Customer Key to Total Customers by right-clicking the measure and selecting Rename.

5. In the Properties window, change the AggregateFunction property for the Total Customers measure from `Sum` to `Count`.

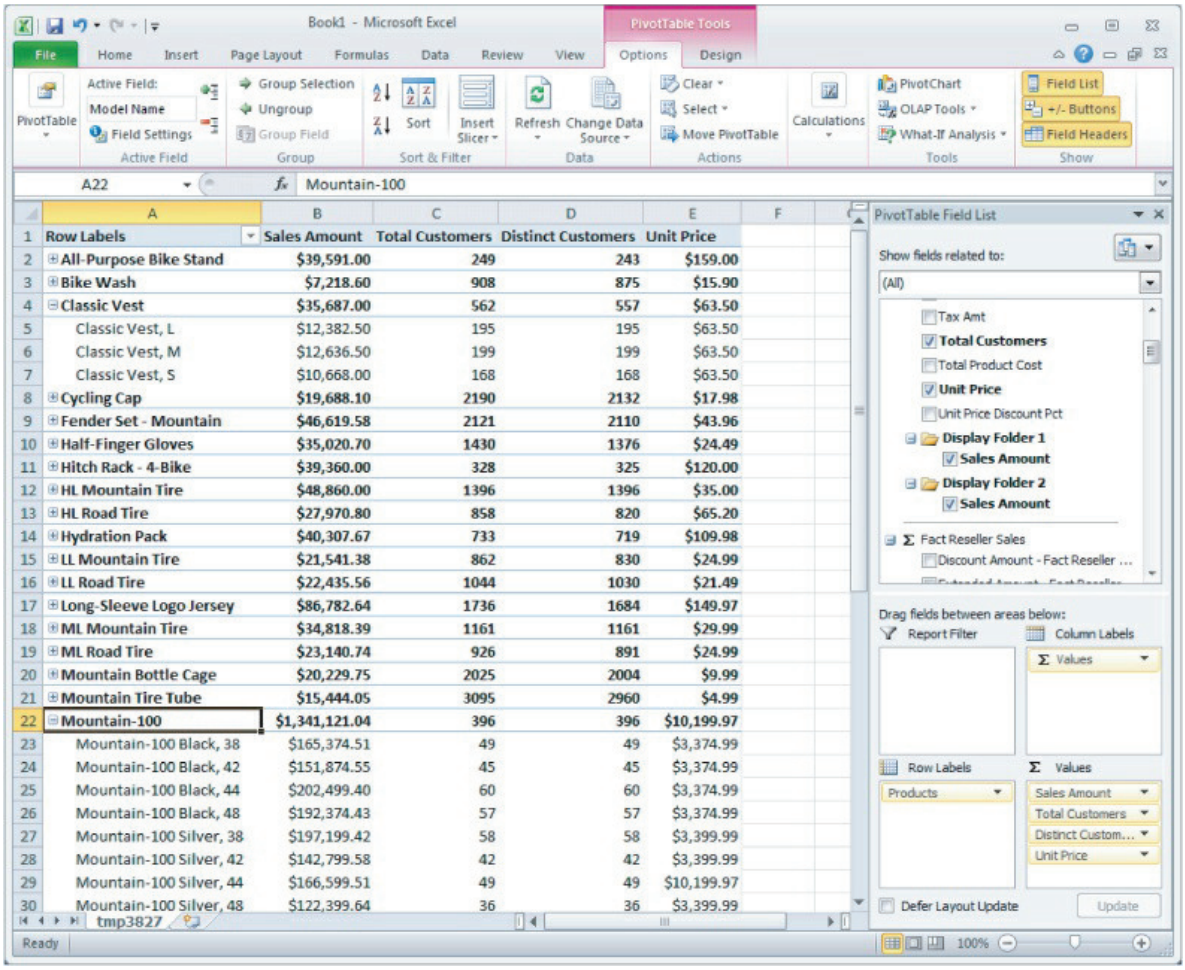
6. The Unit Price of a product is the same value at every level of the hierarchy. Therefore, this value should not be aggregated. To implement this behavior set the Unit Price measure's aggregate function to `FirstNonEmpty`. Also change the property's FormatString property to Currency.

7. Create a new user hierarchy called Products in the Dim Product dimension with two levels, Model Name and English Product Name, where Model Name is the parent level of English Product Name. Rename the English Product Name level to Product Name.

8. Deploy the project to the Analysis Services instance.

9. When the deployment is complete, switch to the Cube Browser tab, and click the Analyze in Excel button on the toolbar. When you look at the Fact Internet Sales measure group in the PivotTable Field List, you see two folders called DisplayFolder 1 and DisplayFolder 2 that contain the Sales Amount measure, as shown in Figure 6.22.

Figure 6.22



10. Drag and drop the Sales Amount, Total Customers, Distinct Customers, and Unit Price measures from the Fact Internet Sales measure group to the Values area of the PivotTable Field List.

11. Then drag and drop the Products hierarchy from the Dim Product dimension to the Row Labels area of the PivotTable Field List. Expand the Classic Vest member.

You can see that the values for the measures are aggregated for the two levels of the Products hierarchy, Model Name and Product Name, based on the aggregate function chosen. Choosing the aggregate functions `Count` and `Distinct Count` not only counts the values for the members of a hierarchy, but also aggregates the counts to the next level. Notice also that the values of Sales Amount and Unit Price are formatted based on the format strings you specified earlier. (Note that in order to see the formatting defined in the Analysis Services database you may have to click on the bottom of the Change Data Source button in the PivotTable Tools on the Options tab of the Excel ribbon, select Connection properties, and in the Connection Properties dialog, check the check boxes in the OLAP Server Formatting section of the dialog.) You can also see that the Unit Price measure is aggregated from the members in the Product Name level to Model Name level based on the `FirstNonEmpty` aggregate function. You see the value for the Unit Price measure for the Classic Vest model as $63.50, the same as the value of each of the specific products aggregated under it, as shown in Figure 6.22. On the other hand, if you expand the Mountain-100 model node, you see that the Products under it have different values. The Unit Price value shown for the Mountain-100 model is that of the Mountain-100 Silver 44 product, one of the members of the Mountain-100 model attribute. This is the value that was chosen by the `FirstNonEmpty` aggregate function.

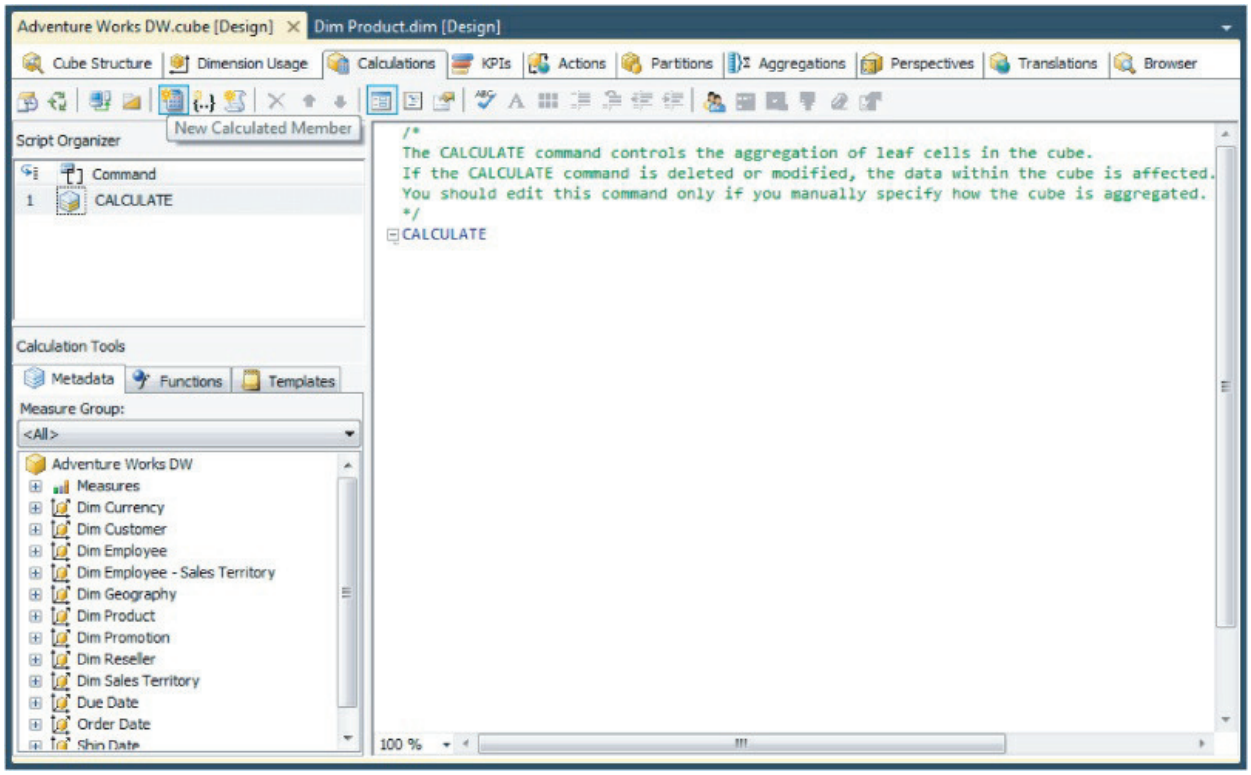
You have now successfully enhanced your cube by adding cube dimensions and measures. In the process you have also learned about properties of cube dimensions, measures, and measure groups. Most often, businesses need complex logic to analyze their relational data. Analysis Services provides you with the ability to embed the complex calculations required for solving business problems in several ways. The most basic operation that every business needs is creating simple arithmetic calculations on base measures or dimension members. Objects created from such calculations are called calculated members.

Calculated Members

The term *calculated member* refers to the creation of any MDX object through a calculation. The calculated member can be part of the Measures dimension where a simple MDX expression such as addition or subtraction of two or more base measures results in a new measure. Such calculated members on the Measures dimension are referred to as *calculated measures*. You can also create calculated

members on other dimensions by specifying an MDX expression. These members are simply referred to as calculated members. To create a calculated member, click the Calculations tab of the cube editor. This takes you to the Calculations page, as shown in Figure 6.23. The Calculations page contains three panes: Script Organizer, Calculation Tools, and Script Editor.

Figure 6.23



The Script Organizer pane shows the names of the calculation objects in the cube. Various types of calculations can be created in the Calculations view such as calculated members and calculated measures. You can also apply a name to a subset of dimension members, which is referred to as a named set. In addition to calculated members and named sets, you can define a script containing MDX expressions that perform complex business logic calculations. If you right-click in the Script Organizer pane, you can see menu items that allow you to create a calculated member, a named set, or a script command. These operations can also be performed using the corresponding toolbar buttons. In this chapter, you create calculated measures. The creation of script commands and named sets are detailed in Chapter 9.

The Calculation Tools pane contains three tabs: Metadata, Functions, and Templates. The Metadata tab shows the measures and dimensions of the current cube. The Functions tab shows all the MDX functions along with a template of the arguments needed for each function. In the Templates tab you can see templates for some common calculations used in certain applications such as budgeting and financial.

The Script Editor pane shows the actual calculation scripts. The default view of the Script window is called the Form View. It presents each calculated member in a form that makes it easy to enter the expression and set properties on the member. You can also switch the Script Editor pane to a different view called the Script View, which shows all the cube calculations in a single page showing the actual MDX for each calculation including the properties that could be specified via controls in form view. When Script View is in effect, the Script Organizer pane is hidden. You can toggle between the two views by clicking the Form View/Script View toolbar buttons or in the main menu using the Cube\Show Calculations command which contains options for Script or Form view.

All commands and selections available in Analysis Services are accessible via keyboard controls. You can switch between the three panes of the Calculations tab of the Cube Designer using the F6 function key or by making the appropriate selection via menu items.

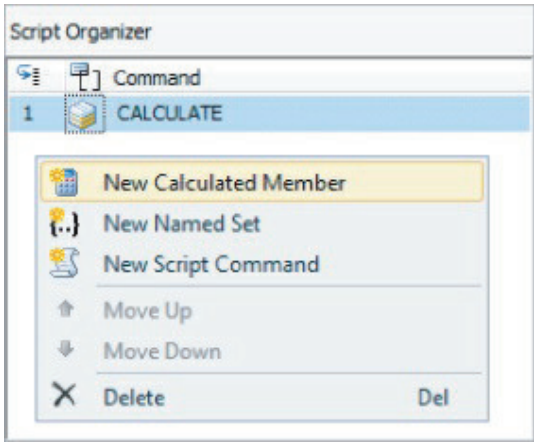
Calculated Measures

Calculated measures are the most common type of calculated members in a cube. In your project you have the Sales Amount and Total Product Cost measures in the two measure groups, Fact Internet Sales and Fact Reseller Sales. An important question to ask about any business concerns profit, which is the difference between total sales and cost of goods sold. In the Adventure Works DW cube, you have Sales through the Internet as well as through resellers. Therefore, you need to add these two sales amounts to calculate the total sales of

products. Similarly, you need to calculate the total product cost by adding the costs of products sold through the Internet and resellers. Two calculated measures must be formed to perform these operations. After you create these two calculations, you can calculate the profit. Follow these steps to create the calculated measure for profit:

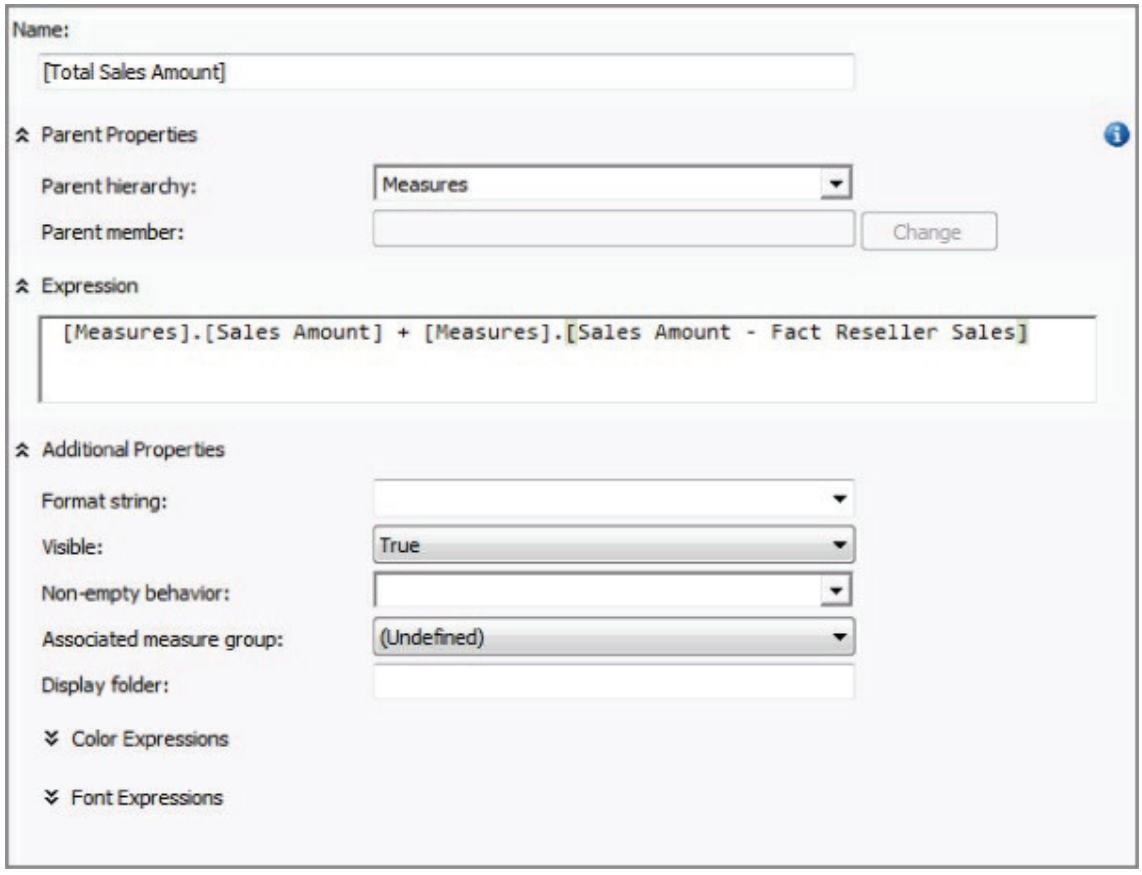
1. Right-click in the Script Organizer pane, and select New Calculated Member, as shown in Figure 6.24. An object called [Calculated Member] is created. The Script window now shows a form with several controls for specifying the name of the calculated member, the MDX expression for the calculation, and other properties.

Figure 6.24



2. Specify the name of the calculated member as [Total Sales Amount] in the Name field of the Script Editor pane. In the Expression text box you need to enter the MDX expression that can calculate the Total Sales Amount. As mentioned earlier, the Total Sales Amount is the sum of sales amounts in the Fact Internet Sales and Fact Reseller Sales measure groups. Drag and drop these measures from the Metadata pane, and add the MDX operator "+" between them, as shown in Figure 6.25.

Figure 6.25

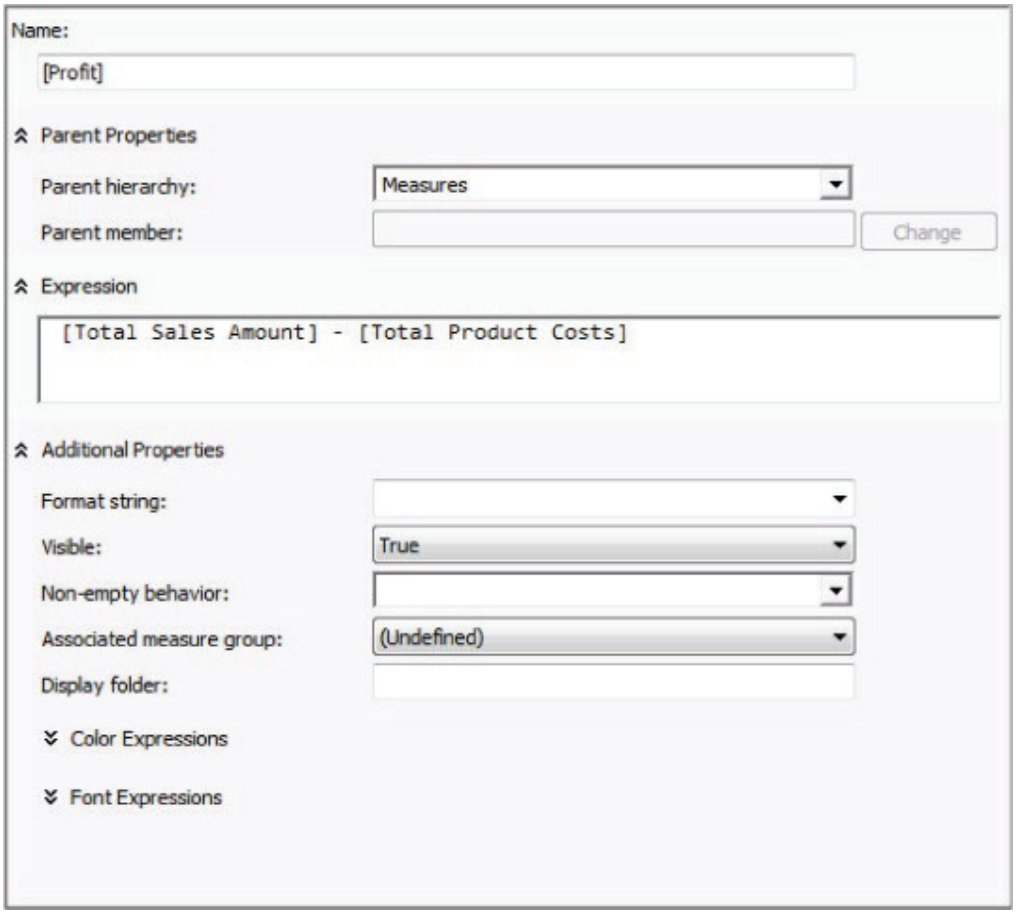


3. For cost of goods sold, create a new calculated member called [Total Product Costs] using a method similar to the one described in step 2 but with appropriate Product Cost measures from the two measure groups.

4. Create a calculated member called [Profit]. The MDX expression to evaluate Profit is the difference of the calculated measures

you have created in steps 2 and 3. Enter the MDX expression `[Total Sales Amount] - [Total Product Costs]` in the Expression text box, as shown in Figure 6.26. Because Measures is a special dimension, you do not need to precede the measure name with [Measures].

Figure 6.26



5. You have the option to specify certain additional properties for calculated measures. By default, all the calculated measures created are visible. You can specify color, font, and format strings for calculated measures based on certain conditions. For example, if you want to highlight the profit in red if the amount is less than one million dollars and in green if it is greater than or equal to one million, you can do so by specifying the appropriate background color for the calculated member.

6. Enter the following MDX expression for the Color Expressions\Back color property in the Script Editor form:

```
iif ( [Measures].[Profit] < 1000000,    255 /*Red*/,
      65280 /*Green*/)
```

This MDX expression uses the `IIF` function. This function takes three arguments. The first argument is an expression that should evaluate to true or false. The return value of the `IIF` function is either the second or the third argument passed to the function. If the result of the expression is true, the `IIF` function returns the second argument; if the expression is false, it returns the third argument. The first argument passed to the `IIF` function is to see if the profit is less than one million. The second and third arguments passed to the function are the values for the colors red and green, respectively. You can also select the values for the color properties by clicking the color icon next to the background color text box.

To see the effect of the calculations you have created, follow these steps:

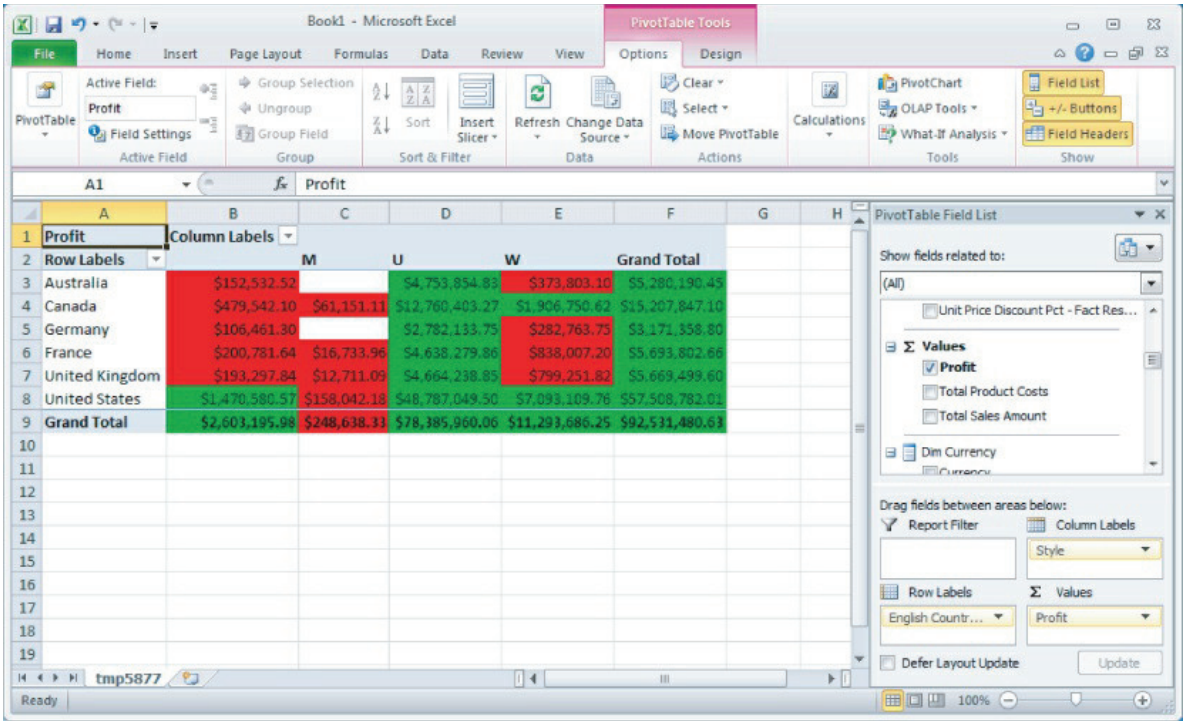
1. Deploy the AnalysisServicesMultidimensionalTutorial project to your Analysis Services instance.

2. To go to the Cube Browser page, and select the Browser tab.

3. When deployment is complete, click the Analyze in Excel button in the toolbar.

4. View the newly created calculated measures in the PivotTable Field List, as shown in Figure 6.27.

Figure 6.27



5. Drag and drop the Profit measure to the Values area, the English Country Region Name attribute of the Dim Geography dimension to the Row Labels area, and the Style attribute of the Dim Product dimension to the Column Labels area. (You must expand the More Fields folder to access the last two items.)

The results are shown in Figure 6.27. Although you can't see color in the figure, on the screen you will see that the background color for cells that contain values are either red or green based on the Profit value.

Querying Calculated Measures

You can query calculated measures similar to other measures in the cube by referencing them by name. For example, if you want to query the calculated member Profit based on Model Name, you execute the following query:

```
SELECT [Measures].[Profit] on COLUMNS,
[Dim Product].[Model Name].MEMBERS on ROWS
FROM [Adventure Works DW]
```

If you want to retrieve all the measures in the cube instead of specifying each measure, use `[Measures].MEMBERS`. However, calculated members are not returned in your query result when you specify `[Measures].MEMBERS`. You need to execute the following MDX query to retrieve the base measures along with the calculated members:

```
SELECT [Measures].ALLMEMBERS on COLUMNS,
[Dim Product].[Model Name].MEMBERS on ROWS
FROM [Adventure Works DW]
```

You have learned to enhance the Adventure Works DW cube by creating calculated measures and learned to set properties for the calculated measures via MDX expressions. The NonEmptyBehavior property for calculated measures is discussed in Chapter 11.

Creating Perspectives

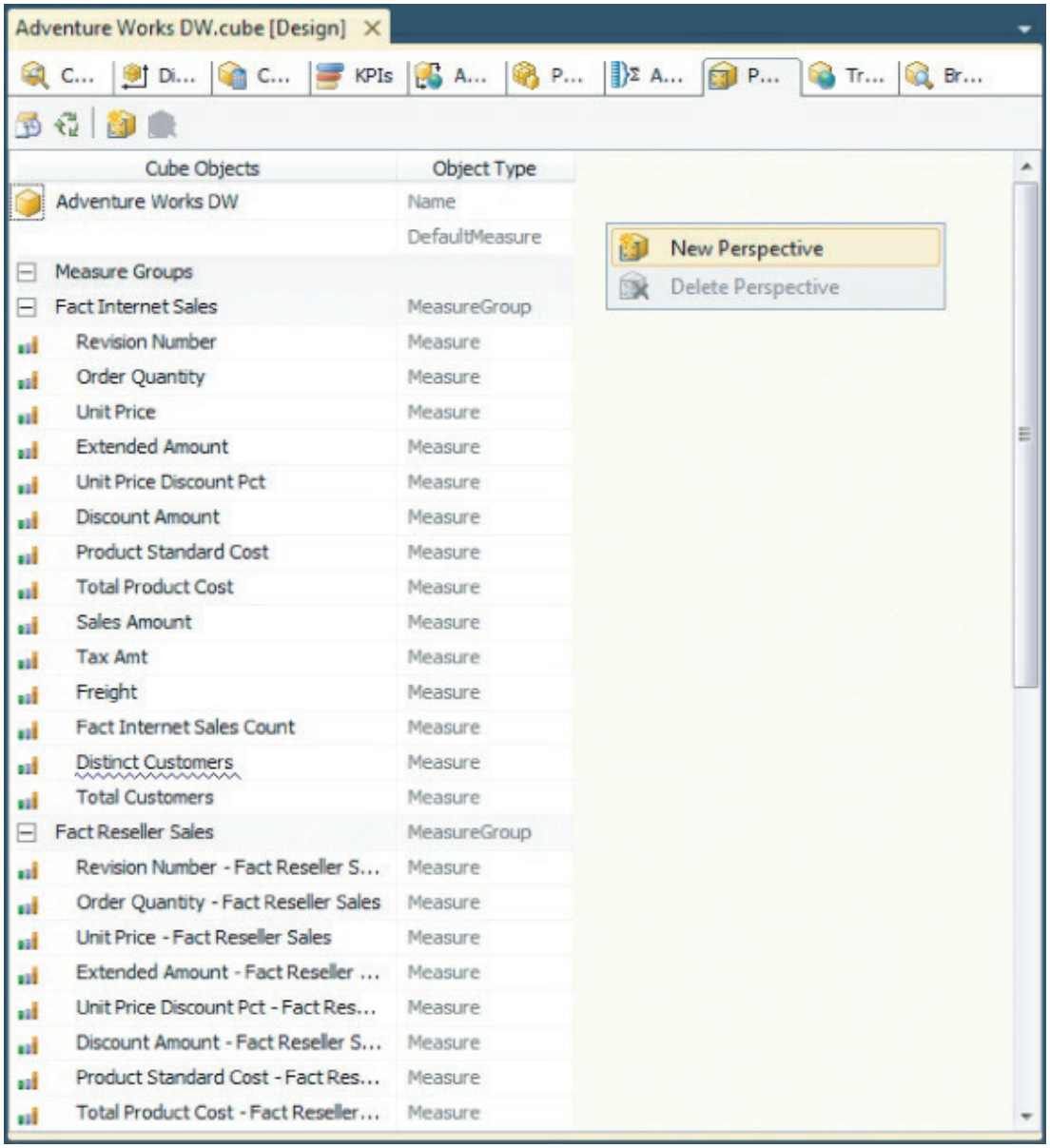
Analysis Services provides you with the option to create a cube that combines many fact tables. Each cube dimension can contain multiple attributes and hierarchies. Even though the cube might contain all the relevant data for business analysis combined into a single object, some users of the cube might be interested only in parts of it. For example, a cube might contain sales and budget information of a company. The Sales department is interested only in viewing sales-relevant data, whereas the users involved in budgeting or forecasting next year's revenue are interested only in budget-relevant sections of the cube. Typically, users do not like to see too much extra information. To accommodate this, Analysis Services allows you to create views of a cube that each contain a subset of all the cube's objects. These views are called *perspectives*.

In the Adventure Works DW cube, you have two fact tables, FactInternetSales and FactResellerSales. To understand the behavior of perspectives, create a perspective for Internet Sales and a perspective for Reseller Sales. The following steps show you how to do this:

1. Click the Perspectives tab in the Cube Designer. You see a column on the left showing the measures, dimensions, and calculated

members in the cube, as shown in Figure 6.28.

Figure 6.28

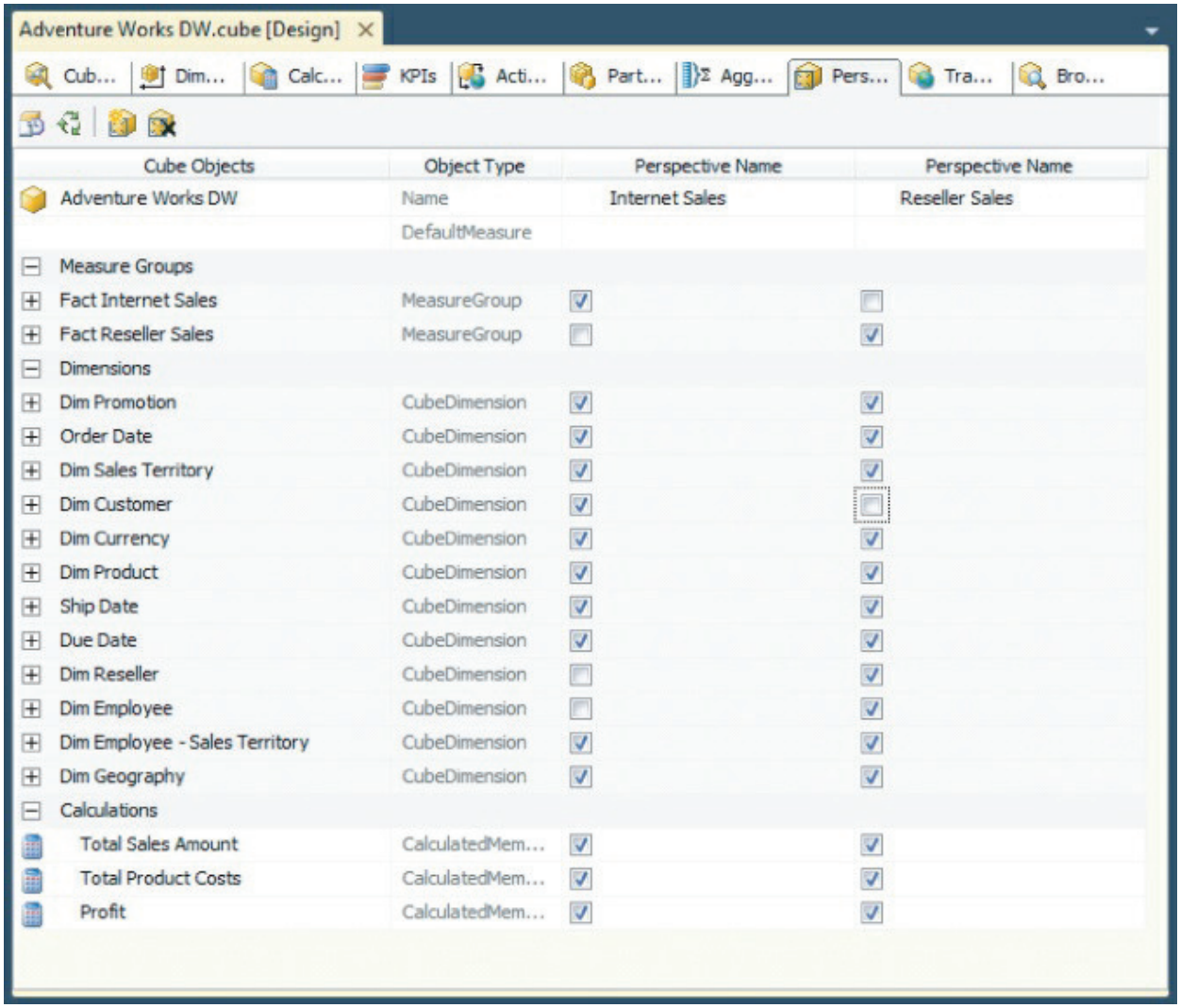


2. Right-click in the Perspectives page, and select New Perspective, as shown in Figure 6.28. You can also create a new perspective by clicking the New Perspective button in the toolbar. A new column with the name Perspective is created. A check box is in the Perspective column next to each object in the cube. Rename the new perspective to Internet Sales. Uncheck the Fact Reseller Sales measure group and the Dim Employee and Dim Reseller dimensions.

3. Create another perspective called Reseller Sales. Uncheck the Fact Internet Sales measure group and the Dim Customer dimension.

Your Perspective window should look similar to Figure 6.29. Now deploy the project. SSDT sends the definitions for the new perspectives to the server. Although perspectives are frequently shown as cubes in client applications such as the SSDT Cube Browser, they are not cubes but rather views of a subset of the objects in an existing cube.

Figure 6.29



In Chapter 5 you learned how to specify translations of attributes and metadata in a dimension. Similarly, you can create translations for metadata of a cube. You see the use of perspectives along with translations after learning how to create translations for a cube.

Creating Translations

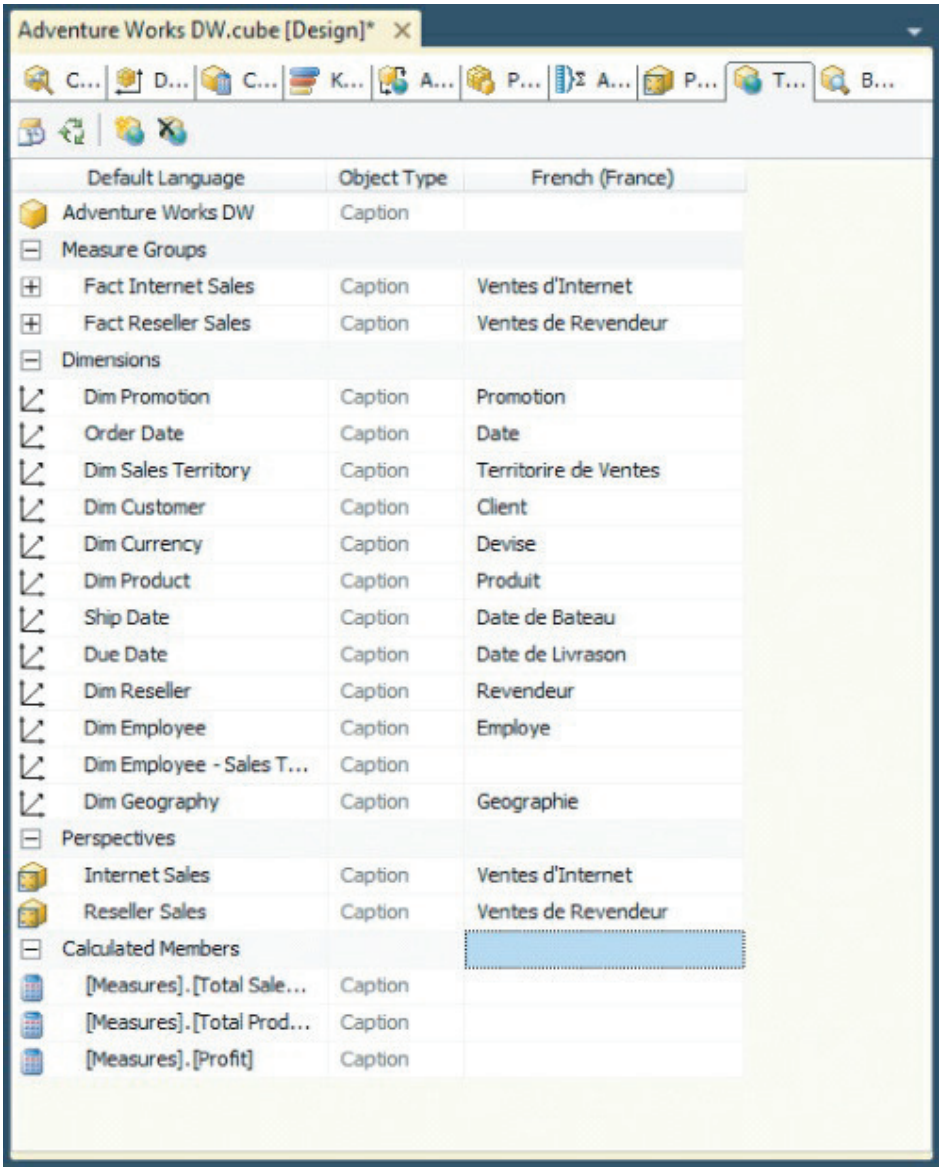
Translations facilitate the display of data and metadata in different languages. Unlike translations for dimensions, where you can specify translations for a dimension's data and metadata, cube translations are specified only for the cube's metadata.

To create a new translation for the Adventure Works DW cube, do the following:

1. Click the Translations tab in the Cube Designer. Similar to the Perspective view, the left column shows the names of all the metadata objects in the default language. There is another column that indicates the object type.

2. Right-click in the Translation page and select New Translation. You can also create a new translation using the New Translation button in the toolbar. In the Select Language dialog, select French (France) as the language, and click OK. You now have a new column where you can provide the translations of each object (measure, display folders, dimension name, attribute names). Specify the translations in French, as shown in Figure 6.30. (If you don't know French, you can enter the translations in a language of your choice.) You can define translations for each metadata object in the cube, such as measure names, measure group names, dimension names, perspective names, as well as calculated member names.

Figure 6.30

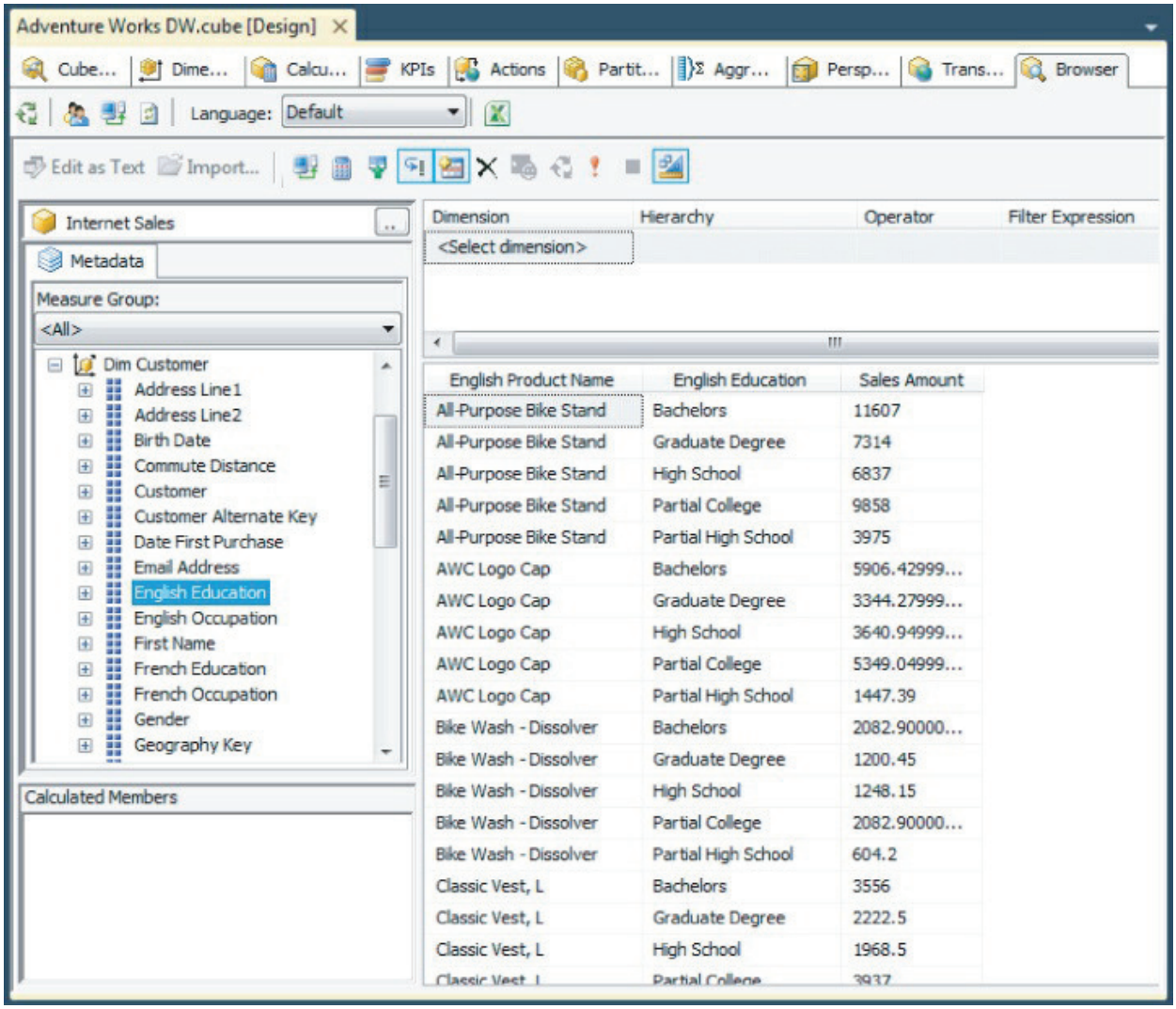


3. Deploy the project to your Analysis Services instance.

Browsing Perspectives and Translations

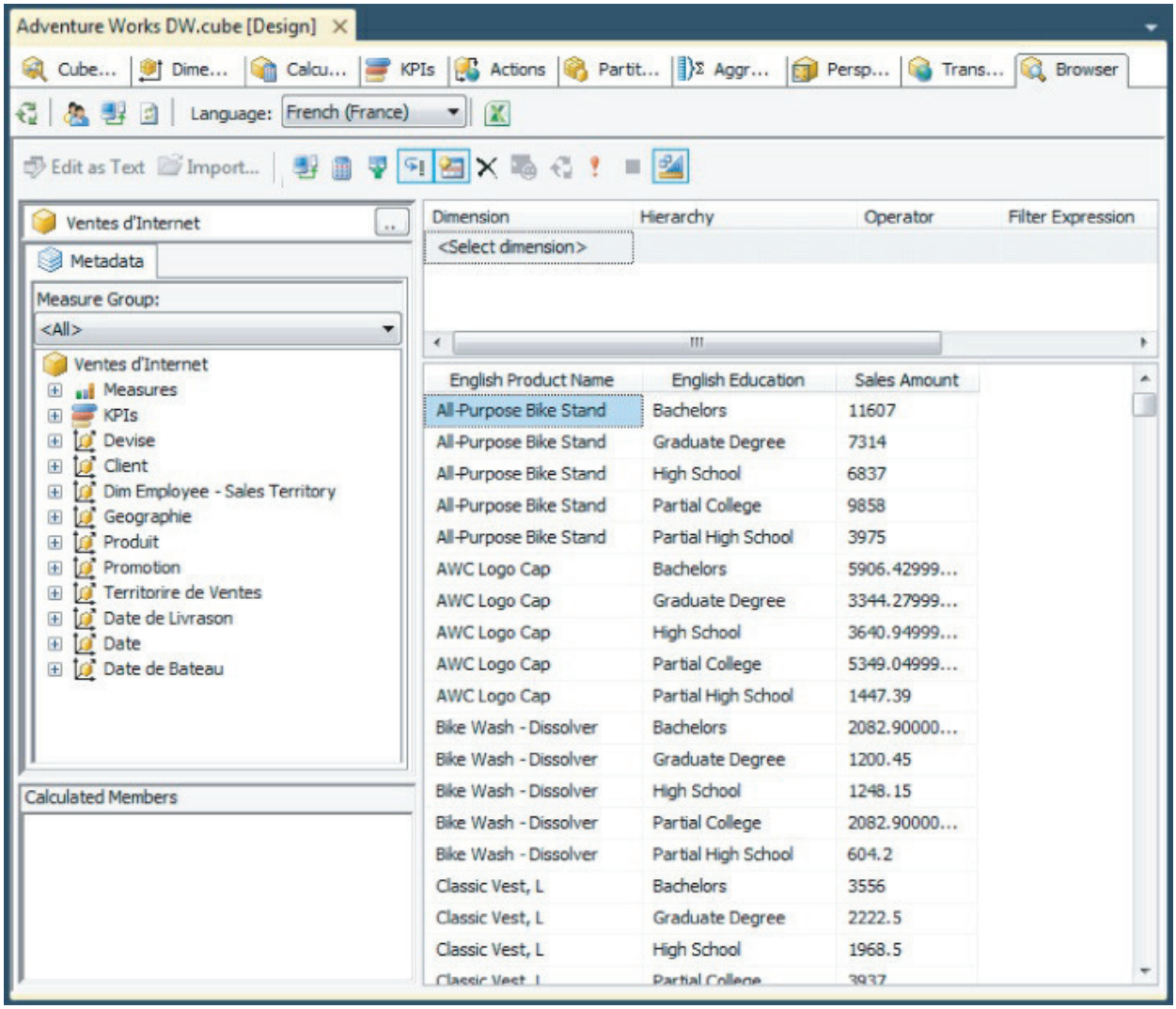
You have successfully created perspectives and translations for the Adventure Works DW cube. You can see these aspects of the cube in the Browser tab of the Cube Designer. In the Cube Browser, if you click the Cube Selection button above the Metadata pane on the left side of the Cube Browser page, you can bring up the Cube Selection dialog that allows you to choose from among three perspectives: Adventure Works DW (the default perspective that contains all cube objects), Internet Sales, and Reseller Sales. Select the Internet Sales perspective and click OK. If you expand the measures in the Measure Group window, all the measures relevant to the reseller are not visible. Drag and drop the Sales Amount, English Product Name attribute from the Dim Product dimension, and the English Education attribute of the Dim Customer dimension to the Report pane of the Cube Browser. You can see the sales amount data along with product names and education of customers, as shown in Figure 6.31.

Figure 6.31



To see the translated names in French, select the French (France) language in the Language drop-down at the top of the Cube Browser. When you select the French (France) language, the metadata and data members for the hierarchies in the Cube Browser automatically change to values in French for those objects where translations have been defined, as shown in Figure 6.32. Thus, you have created translated values in French for a French client who wants to analyze the same values, but who can understand and interpret the results better in French than English. Each language has a corresponding ID called the locale ID. When you select a specific language in the Browser, SSDT connects to the server specifying the corresponding locale ID for the selected language. Analysis Services returns the metadata and data corresponding to the locale ID when queries are sent to the server on the same connection.

Figure 6.32



Instead of creating separate cubes for various users and clients understanding different languages or wanting to see subsets of the main cube you've built along with the overhead of maintaining those cubes for changes that need to be made, Analysis Services Translations and Perspectives features enable you to satisfy those types of customer needs with one cube.

Summary

You traversed the Cube Wizard for a second time in this book, but at a different level of granularity and hopefully with more understanding of what occurs. You learned how to create calculated members and set properties concerning the display of those members, for example, different color foregrounds and backgrounds. And finally, you learned how to create and browse perspectives and translations. In the real world of business, you have additional enhancement requirements to meet after running the Cube Wizard. These requirements may include creating calculated members on dimensions, creating cube scripts containing complex MDX expressions to meet your business needs, and adding Key Performance Indicators (KPIs), which can graphically represent the state of your business in real time. The Cube Designer contains additional tabs for KPIs and Actions. These features help enhance your cubes for business analyses. In addition, the Cube Designer helps in partitioning fact data and defining aggregations, which in turn help you achieve improved performance while querying your multidimensional model. These are covered in Chapter 9, with additional coverage in other chapters as well. In the next chapter you learn how to manage your Analysis Services databases using the SQL Server Management Studio.