Лекция 6

Тема «Working with Data Sources and Data Source Views »

Working with Data Sources and Data Source Views

You have completed the first three chapters of the book where you learned the concepts of data warehousing, worked hands on with SQL Server Analysis Services (SSAS) 2008 tools, and finally 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 designing your dimensions and cubes is based upon an existing single - source data set. In the real world, you will be working with multiple relational data sources when you develop business intelligence applications. In this chapter you learn what Data Sources are and how they feed into the creation of Data Source Views (DSVs). These DSVs provide you 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. Note that more than one data source per project is supported as are multiple DSVs per project. You learn how this infrastructure plays out in this chapter.

Data Sources

In order to retrieve data from a source you need information about the source such as the name of the source, the method used to retrieve the data, security permissions needed to retrieve the data, and so on. All this information is encapsulated into an object called a *Data Source* in SSAS 2008. An Analysis Services database contains a collection of Data Sources, which stores all the data sources used to build dimensions and cubes within that database. Analysis Services will be able to retrieve source data from data sources via the native OLE DB interface or the managed .NET provider interface.

In the simplest case you will 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.

A variant on the data source usage, which is enabled by the UDM first introduced in Analysis Services 2005, is the ability to take data directly from the OLTP system as input to the BI application. This is shown in Figure 4 - 2 .

OLTP
System
(Operational Data
Source)
ODS
(Operational Data
Store)
BI Application
(Data Source
Consumer)

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

Figure 4-1

OLTP
System
(Operational Data
Source)
ODS
(Operational Data
Store)
BI Application
(Data Source

Consumer)
Data Sources enabled with UDM; 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.
Figure 4-2

Prior to the introduction of the Analysis Services UDM, certain limitations were associated with the use of data sources. Pre - UDM versions of Analysis Services only supported one fact table per cube. Therefore, only one data source could be used for specifying the fact table of a cube. You could still specify multiple data sources within Analysis Services in those earlier versions because dimensions referenced did not have to be in the same data source as the fact table but the fact table data had to come from a single data source. A workaround addressing the single fact table constraint was to create a SQL view on the multiple fact tables to create what appeared to Analysis Services as a single fact table. A more common and straightforward solution adopted by many users was to have multiple cubes based on disparate data sources and combine them into a single cube that was called a *virtual cube* .

With the UDM, Analysis Services now natively supports the capability of specifying multiple fact tables within a single cube. Each of these fact tables can be from a different data source. Analysis Services 2008 provides you with the capability of creating what are essentially virtual cubes; this is accomplished using linked objects (discussed in Chapter 9). Because Analysis Services 2008 provides you with the capability of creating cubes from various data sources, you need to be extremely careful about how you model your cube — that is, you must specify the right relationships (primary key and foreign key mappings) between tables from various data sources. In this way you can make sure your cube is designed to provide you the results you want.

Using the pre - UDM data source method is like carving on a bar of soap with a butter knife: You could create a statue, but it might not win any awards for beauty. Conversely, the kind of power and flexibility in Analysis Services 2008 puts you in a position similar to that of carving a bar of soap with a razor blade. Carving with a razor blade, you can make a gorgeous and intricate statue, but if you ' re not careful, you could cut the heck out of your fingers. So, be careful and craft some beautiful dimensional schemas! To do so, keep your schemas as simple as possible relative to the flexibility requirements imposed by the application specification you ' re working with.

Data Sources Supported by Analysis Services

Strictly speaking, Analysis Services 2008 supports all data sources that expose a connectivity interface through OLE DB or a .NET Managed Provider. This is because Analysis Services 2008 uses those interfaces to retrieve the schema (tables, relationships, columns within tables, and data types) information it needs to work with those data sources. If the data source is a relational database, then by default it uses the standard SQL to query the database. Analysis Services uses a cartridge mechanism that allows it to use the appropriate SQL language and extensions to talk to different relational database systems.

Analysis Services 2008 officially supports specific relational data sources. The major relational data sources for Analysis Services databases include Microsoft SQL Server, IBM ' s DB2, Teradata, Oracle, Sybase, and Informix. Figure 4 - 3 shows various data sources supported by Analysis Services 2008 on one of the machines that has SQL Server 2008 installed. For a specific data source you need to install the 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 supported on your development machine where you use the Business Intelligence Development Studio (BIDS) to design your database, but also on the server machine where an Analysis Services instance will be running. For relational databases DB2 and Oracle it is recommended you use the Microsoft ' s OLE DB data provider for Oracle or DB2 instead of the OLE DB providers provided those databases. Please make sure appropriate connectivity components from Oracle and IBM ' s DB2 are installed on your machine in addition to the OLE DB providers from Microsoft.

In Chapter 2 you used the Data Source Wizard to create a data source that included impersonation information. We will use the AnalysisServices2008Tutorial 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 of creation of the data source. Alternatively you can define additional connection properties after the creation of the data source by double - clicking the created data source and 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 the relational data sources SQL Server and Oracle, is used to ensure that the data read by SSAS 2008 is consistent across multiple queries sent over a single connection. What this means is that if the data on the relational data source keeps changing and if multiple queries are sent by SSAS 2008 to that relational data source, all the queries will be seeing the same data seen by the first query. Any changes to the data between the first query and Nth query sent over a specific connection will not be included in the results of the first through Nth query. All the specified connection properties will be stored and applied whenever a connection is established to that specific data source. The Data Source Wizard also allows you to create data sources based on an existing data source connection already created so that a single connection can be shared by Analysis Services for multiple databases. The wizard also allows you to establish connections to objects within the current Analysis Services project, such as establishing an OLE DB connection to the cube being created in the project. Such a connection is typically useful while creating mining models (discussed in Chapter 16) from cubes.

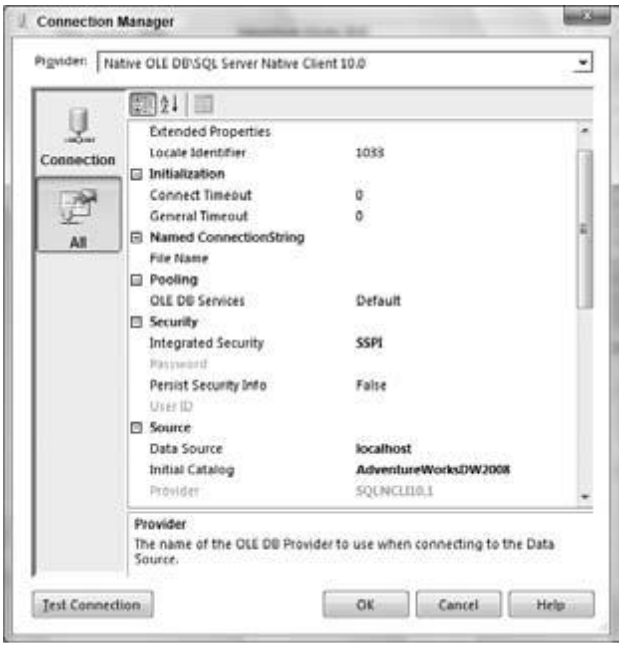


Figure 4-4

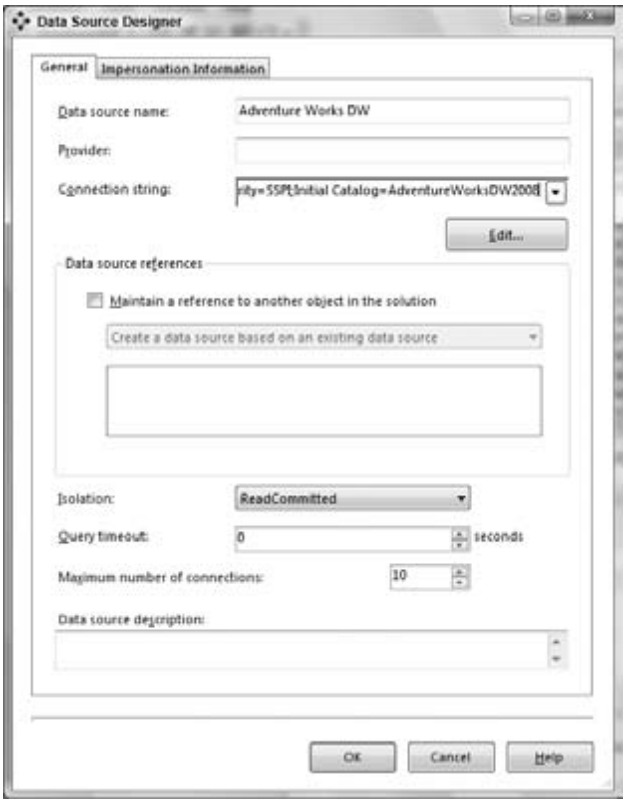


Figure 4-5

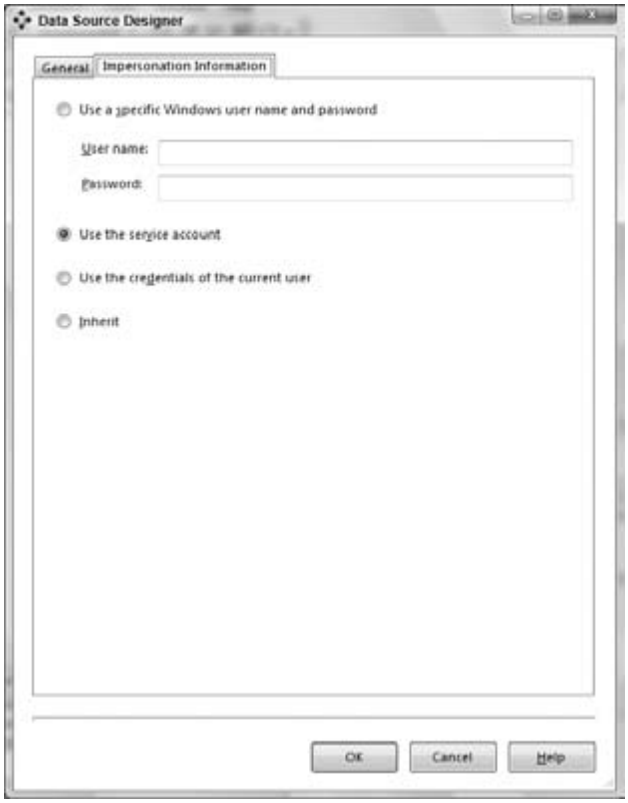
The Impersonation Information dialog in the Data Source Wizard has four options to choose from as shown in Figure 4 - 6 . You briefly learned about these options in Chapter 2 . At development time, BIDS uses the current user ' s credentials to connect to the relational backend and retrieve data. However, after the Analysis Services project is deployed, Analysis Services needs credentials to connect to and retrieve data. You specify the impersonation information so that Analysis Services can use the right credentials to

make that connection. The following lists more details on the four options and when each option is likely to be used:

Use a specific Windows user name and password: You typically would choose this option when the SSAS instance service startup account does not have permissions to access the relational backend. When you select this option you need to specify a Windows username and password that SSAS will use to connect to the relational backend. Due to security reasons the username and password are encrypted and stored. Only the encrypted password is sent to the SSAS instance when the project is deployed.

Use the service account: This is the option typically selected by most users. You need to make sure the service startup account of the SSAS instance has access to the relational backend.

Use the credentials of the current user: This option is typically selected for data mining. This option can be used for out - of - line bindings, DMX OPENQUERY statements, local cubes, and mining models. Do not select this option when you are connecting to a relational backend for processing, ROLAP queries, remote partitions, linked objects, and synchronization from target to source.



. NET versus OLE DB Data Providers

There are two types of data providers that most data sources support. OLE DB defines a set of COM interfaces that let you access data from data sources. There is also a .NET managed code interface similar to OLE DB. Providers implementing that interface are called .NET providers. SSAS 2008 has the ability to use OLE DB or .NET providers to access data from data sources ranging from flat files to large - scale databases such as SQL Server, Oracle, Teradata, DB2, Sybase, and Informix. SSAS retrieves data from the data sources using the chosen provider ' s (OLE DB or .NET) interfaces for processing of 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. Updating the data in the UDM is called writeback. Analysis Services also uses the provider interfaces to update the source data during writeback (you learn about writeback in Chapter 12).

. NET Providers

Microsoft has created the .NET Framework and programming languages that use the framework to run in the Common Language Runtime (CLR) environment. The relationship between the Microsoft languages and the CLR are analogous to that of Java the language and the Java Runtime (the virtual machine). The .NET Framework itself is a huge class library that exposes tons of functionality and does

so in the context of managed code. The term *managed* refers to the fact that memory is managed by the CLR and not the coder. You can write your own managed provider for your data source, or you can leverage .NET providers that use the .NET Framework. With the installation of SQL Server 2008 you will have .NET providers to access data from Microsoft SQL Server and Oracle as shown in Figure 4 - 3 . If your relational data source 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 the .NET provider to connect to your data source.

OLE DB Data Providers

OLE DB is an industry standard that defines a set of COM (Component Object Model) interfaces that allow clients to access data from various data stores. 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. These OLE DB providers are delivered together in a package called MDAC (Microsoft Data Access Components). 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 back - end 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 has been the primary way of connecting to a Microsoft SQL Server prior to the release of SQL Server 2005. From the SQL Server 2005 release, this OLE DB provider has been repackaged and named SQL Server Native Client. SQL Server Native Client provides easy manageability of upgrades to the OLE DB provider. The SQL Server 2008 release provides version 10 of SQL Server Native Client and is used in the data source connection string as shown in Figure 4 - 5 . SSAS 2008 provides the capability of connecting 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. SSAS 2008 and SSAS 2005 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, we recommend you 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 really concerned about the fastest possible performance, we recommend you use OLE DB providers for your data access.

Data Source Views

Data Source Views (DSVs) enable you to create a logical view of only the tables involved in your data warehouse design. In this way, system tables and other tables not pertinent to your efforts are excluded from the virtual workspace. In other words, you don ' t have to look at what you ' re never going to use directly anyway. DSVs are a powerful tool. In fact, you have the power to create DSVs that contain tables from multiple data sources, which you learn about later in this chapter. You need to create a DSV in your Analysis Services database because cubes and dimensions are created from a DSV rather than directly from the data source object. The DSV Wizard retrieves the schema information including relationships so that joins between tables are stored in the DSV. These relationships help 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, we recommend you create them within the DSV. Defining the relationships between the tables in the DSV helps you to get a better overview of your data warehouse. Taking the time to create a DSV ultimately pays for itself in terms of speeding up the design of your data warehouse.

Back in Chapter 2 you used the DSV Wizard to create a view on the Sales fact tables in Adventure Works DW. The DSV Wizard is a great way to get a jump - start on DSV creation. Then, once the DSV is created, you can perform operations on it such as adding or removing tables, specifying primary keys, and establishing relationships. These operations are accomplished within the DSV Designer. You learn more about the DSV Wizard and the DSV Designer and the operations within them in the following sections.

DSV Wizard

In Chapter 2 , the DSV Wizard helped you create a DSV by going through a few dialogs. You need a data source to create a DSV. If you had not created a data source object in your database, the DSV Wizard allows you to create new data sources from the DSV Wizard ' s Select a Data Source page by clicking the New Data Source button. In addition, the DSV Wizard also allows you to restrict specific schemas as well as filter certain tables, which helps you to work with only the tables you need to create your DSV.

DSV Designer

The DSV Designer contains three panes, as shown in Figure 4 - 7 . The center pane contains a graphical view of all the tables in the DSV and their primary keys. The relationships between tables are represented by lines with an arrow at the end. The top - left pane is called the *Diagram Organizer* , which is helpful in creating and saving concise views within large DSVs. When a DSV contains more than 20 tables it is difficult to visualize the complete DSV in the graphical view pane. When there are a large number of tables you will likely perform operations only on a subset of these tables at any given time. The Diagram Organizer is a handy way to create several diagrams that include just such subsets of relevant tables. Note that operations done on the tables within this diagram are reflected real - time in the entire DSV. By default you get one diagram that is called All Tables and contains the entire DSV.

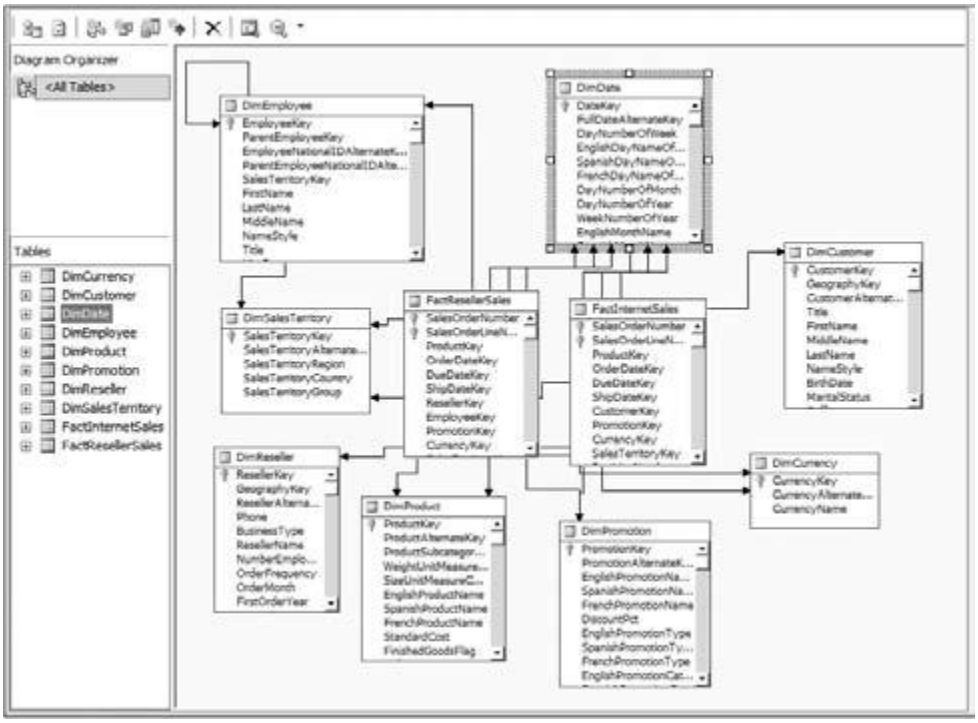


Figure 4-7

Figure 4 - 7 shows part of the default diagram All Tables that is created at the completion of the DSV Wizard. The lower - left pane of the DSV Designer is called Tables and is used to show the tree view of all the tables of 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 indicates all 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 will see that the DimCurrency table joins to the FactInternetSales and FactResellerSales tables through the CurrencyKey — where the join column is specified within parentheses.

Figure 4-8

Adding/Removing Tables in a DSV

It is most common to initially create DSVs using the DSV Wizard. Also common is the desire to modify what the wizard generates to maximize the usefulness of the view. What the wizard generates is usually good, but subject to improvements. The DSV Designer provides you with the capability to easily modify the DSV. To modify the existing tables, right - click the diagram view pane and select Add/ Remove Tables, as shown in Figure 4 - 9 .

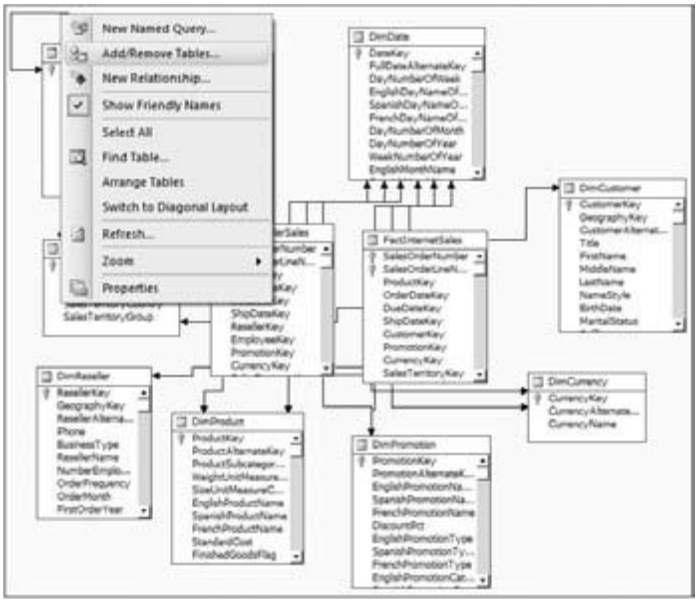


Figure 4-9

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 from the DSV in the DSV Designer in the graphical view pane or the table view pane using the following steps:

1. Select the table to be deleted.
2. Right - click the table and click Delete Table from DSV.
3. Click OK in the confirmation dialog that appears.

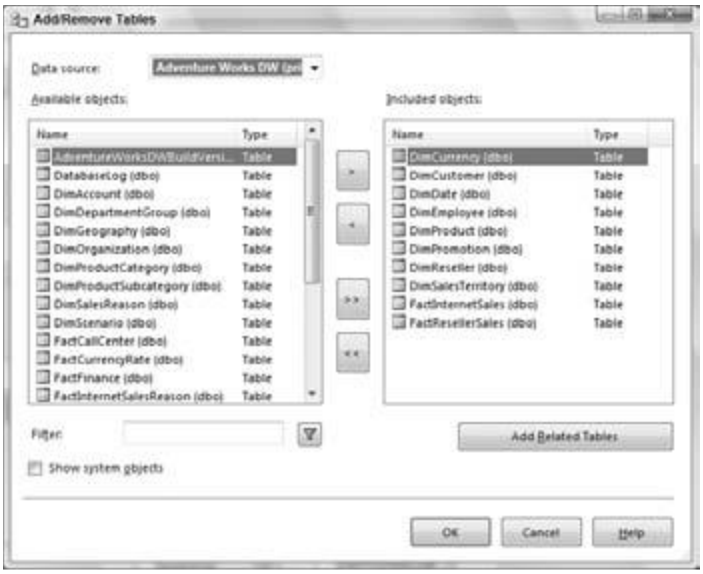


Figure 4-10

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 will need in place for preparation of data for analysis — that is, for building dimensions and cubes. The DSV Wizard extracts the primary keys and the relationships specified in the underlying relational database to form primary keys and the relationships represented in the DSV. But perhaps some of the OLTP systems you use do not have the primary keys and relationships specified in the relevant tables — or when you design your data warehouse you might want to change these to suit your data warehouse design. The DSV Designer provides you with the functionality to specify primary keys for the tables that do not have them already, and in this way you can effectively modify or add new relationships between the tables in the DSV.

To specify the primary key(s) for a table, you need to do the following in the DSV Designer:

1. Select the column in the table that you want to specify as a 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. If the tables have auto - increment setup for the key column in the database, you will not be able to change the primary key(s) of the tables.
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 that are involved in the join. Typically, column B is the primary key in Table2. Column A is referred to as the foreign key. An example would be a Sales fact table that has a Product ID as a column that joins with the Product ID column in the Products dimension table. In order to specify relationships between tables in the DSV, you use the following steps:
3. Select column A in Table1 that is involved in the join to Table2.
4. 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 will see details on the relationship — the tables involved in the relationship and the columns used for the join. Figure 4 - 11 shows the relationship between the FactResellersSales 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 that are involved in the join.

You can also create a new relationship by right - clicking a table and selecting New Relationship. You will be asked to specify the relationship in the Create Relationship dialog, which is similar to the Edit Relationship dialog shown in Figure 4 - 11 . You need to choose the columns in the source and destination tables that are involved in the join.

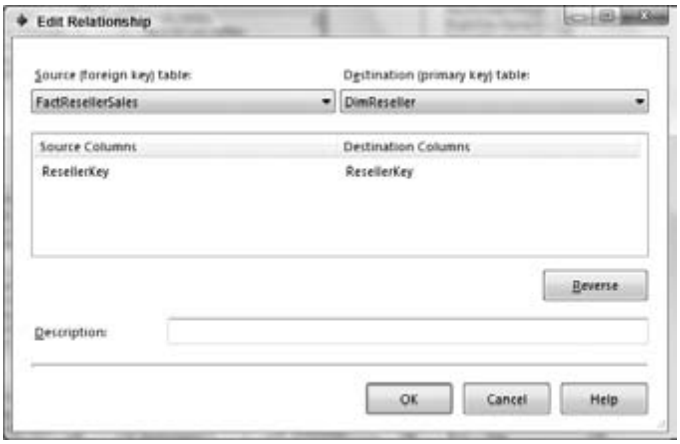


Figure 4-11

All graphical operations such as drag-and-drop and specifying primary keys that are accomplished in the diagram view can also be accomplished in the table view.

Customizing Your Tables in the DSV

While modeling your data warehouse you will often want to select a few columns from tables, 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. All these operations can be done by creating views in the relational database. SSAS 2008 provides the functionality of performing all these particular operations within the DSV using a *Named Query* . You can invoke the Named Query editor by right - clicking a table and selecting Replace Table With New Named Query, as shown in Figure 4 - 12 . If you want to add a specific table twice in your DSV or add some columns of a new table, you can launch the query designer by right - clicking the DSV Designer and selecting With New Named Query.

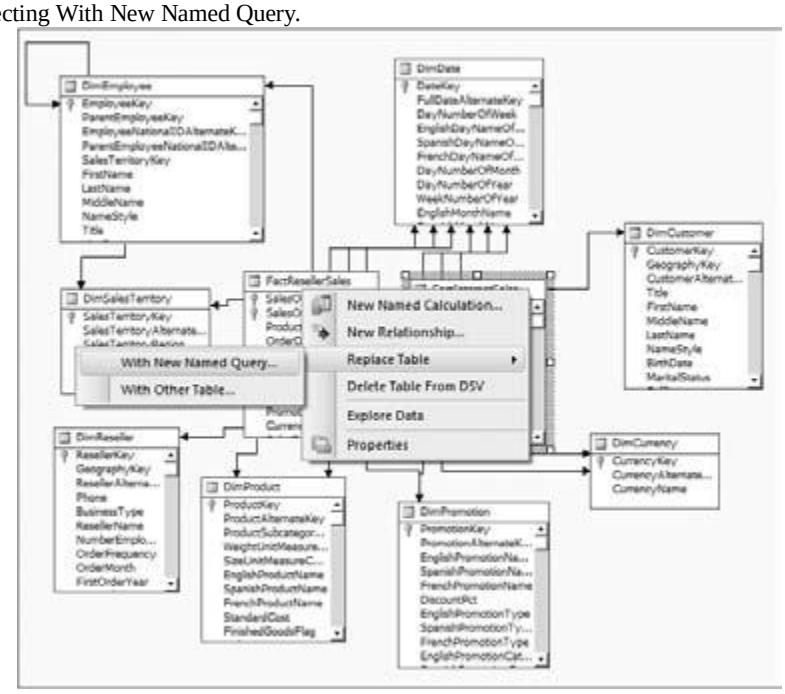


Figure 4-12

Named Queries are created using a query designer that helps you build custom queries to create a view. The Create Named Query designer dialog is shown in Figure 4 - 13. The Named Query editor in the designer is Visual Studio ' s Visual Database Tools query (VDT) editor. This shows the tight integration SQL Server 2008 has with Visual Studio 2008. In this dialog you can add tables from the data source, select specific columns from the tables, and apply restrictions or filters using the graphical interface.
A SQL query is created based on your selections and is displayed in the SQL pane in the editor. If

you ’ re a SQL wizard, you can forego filling out the dialog elements and enter or paste a valid SQL query directly into the SQL pane. We recommend that you then execute the query to make sure the query is correct. The results from the underlying relational database will then be visible in a new pane beneath the SQL pane. Click OK once you have formed and validated your query. The table is now replaced with results from the Named Query you have specified in the DSV.

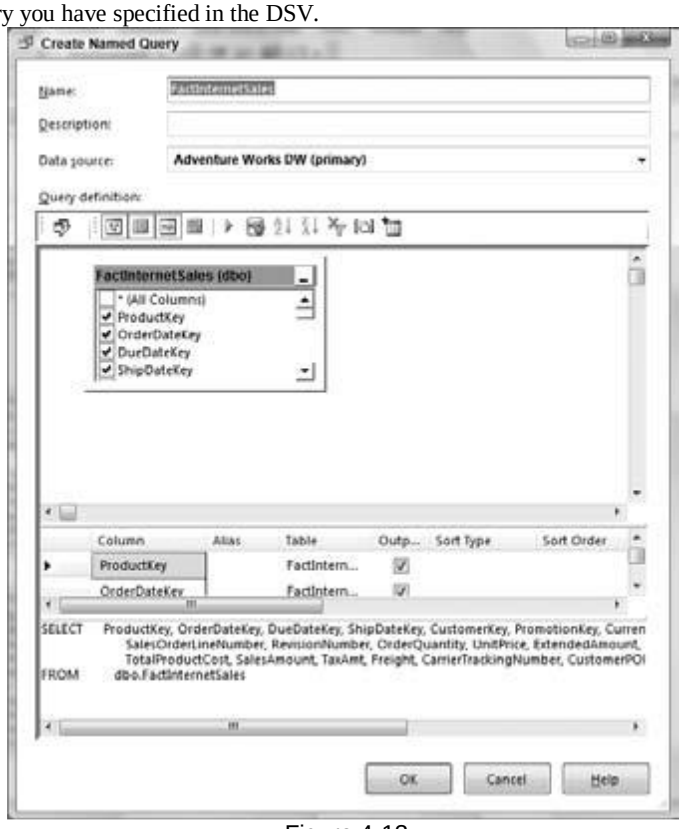


Figure 4-13

In certain instances you might want to create a new column in the table. An example of this would be to create the Full Name of an Employee from the first name, middle initial, and last name. One way to accomplish this task would be to replace the table with a named query and write the appropriate SQL to create this additional column. However, SSAS 2008 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 shown in Figure 4 - 15 . To add a column called Full Name to the Employee table you just need to combine the first name, middle name, and last name. You can type the expression for this in the Expression pane as shown in Figure 4 - 15 and then click the OK button.

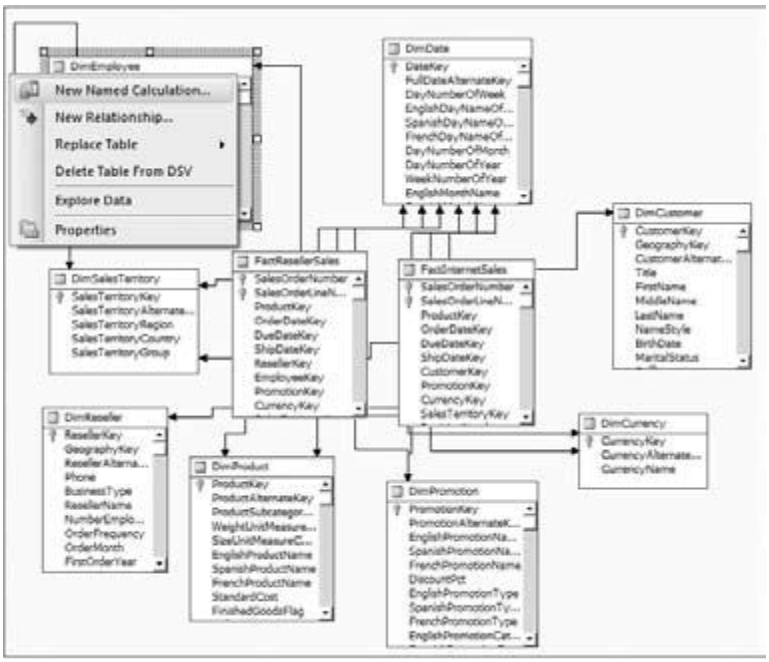


Figure 4-14

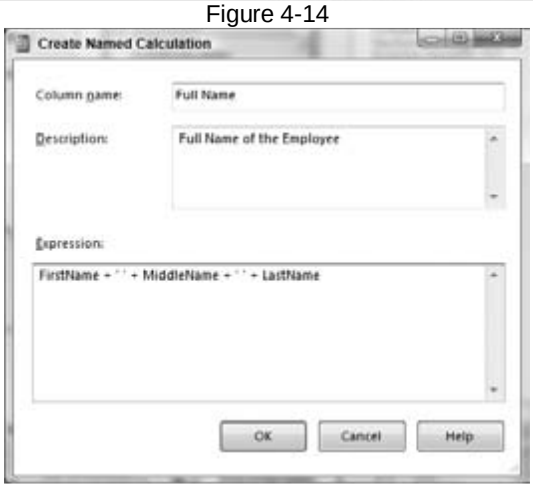


Figure 4-15

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 will be an integer. In the preceding example the data type of this column is a string.

The DSV maintains the calculated column of a table as a computed column in the metadata; 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 must be added to the SQL query so that you can see the data of this computed column.

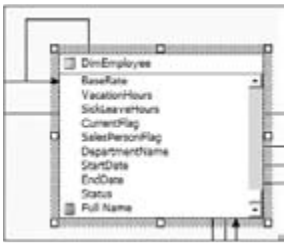


Figure 4-16